

SmartBadge: An Electronic Conference Badge using RF and IR Communications

A thesis submitted in partial fulfilment
of the requirements for the Degree of
Master of Engineering
in Electrical and Computer Engineering
from the University of Canterbury
Christchurch, New Zealand

Mark Alexander White
B.E. (Hons)

February 2006

Abstract

This thesis describes the design and development of the SmartBadge; an electronic replacement for the standard paper name badge worn at conferences and similar events. Both hardware and software have been designed for the SmartBadge; the hardware has been developed around a CC1010 microcontroller and RF transceiver. Attached to this are an infrared transceiver, an LCD display, some LEDs, buttons and a piezoelectric buzzer. There is also an antenna for the RF transceiver whose design is the result of SuperNEC [1] simulations.

Protocol software development has focussed on the communication between a SmartBadge and other badges and base stations, yet there is still space available in the CC1010s flash memory to develop applications beyond the business card exchange example developed to demonstrate the communication software.

The SmartBadge communicates with other badges by using the infrared transceiver. In the business card application a SmartBadge is worn by a person and is collecting the ID and a time counter from SmartBadges worn by other facing people as this person mingles through a conference or similar event. This data is then collected in real time using the RF transceiver to communicate with base stations which would be scattered around the venue.

The RF network has been designed as a single hop network and a new Medium Access Control (MAC) protocol has been designed to allow the SmartBadges to share the links to the base stations while conserving as much energy as possible. This protocol is called Uplink MAC (or U-MAC) and is described in section 6.2.

Acknowledgements

Firstly I would like to thank my supervisors, Professor Desmond Taylor and Dr Mark Billingham, for all the help, guidance and support they have given me on this project.

I would also like to thank the rest of the staff and students at the HITLabNZ (Human Interface Technology Laboratory New Zealand) for providing me with support and a great place to work.

Thanks to the staff in the Electrical Workshop too (Dermot Sallis, Scott Lloyd and Nick Smith) for all their help in the construction of my SmartBadge.

Also thanks to Peter Green for helping me to debug the RF section and showing me how to use all the relevant test equipment (RF test set and Network Analyser).

And finally a big thank you to all my family and friends for their support over the past couple of years.

Mark White

February 2006

Contents

ABSTRACT.....	I
ACKNOWLEDGEMENTS	III
CONTENTS.....	V
LIST OF FIGURES	IX
LIST OF TABLES	XI
 CHAPTER 1 INTRODUCTION.....	 1
1.1 MOTIVATION	1
1.2 DESIGN GOALS	2
1.3 THESIS CONTRIBUTIONS	2
1.4 THESIS STRUCTURE.....	3
 CHAPTER 2 PREVIOUS WORK.....	 5
2.1 INTELLIGENT BADGES	5
2.1.1 <i>Thinking Tag, Meme Tag and nTag</i>	5
2.1.2 <i>CharmBadge</i>	7
2.1.3 <i>Pendle</i>	8
2.1.4 <i>RFID Badges</i>	8
2.1.5 <i>Comparison of Badge Features</i>	9
2.2 CC1010 VERSUS OTHER TECHNOLOGIES	10
2.2.1 <i>CC1010 versus Bluetooth®</i>	10
2.2.2 <i>CC1010 versus RFID</i>	11
2.3 MAC PROTOCOLS	11
2.3.1 <i>Sensor MAC (S-MAC)</i>	12
2.3.2 <i>S-MAC Based Protocols</i>	14
2.3.2.1 Timeout MAC (T-MAC)	14
2.3.2.2 Wireless Sensor MAC (WiseMAC)	16
2.3.2.3 Mobility aware Sensor MAC (MS-MAC)	17
2.3.2.4 Other Protocols.....	18
2.4 CHAPTER SUMMARY	19
 PART I: HARDWARE.....	 21
 CHAPTER 3 ANTENNA DESIGN	 23
3.1 DESIGN GOALS	23
3.2 BACKGROUND	24
3.3 PROTOTYPE ANTENNA	24
3.3.1 <i>Results</i>	26
3.4 FINAL ANTENNA DESIGN	29
3.4.1 <i>Antenna Impedance</i>	30
3.4.1.1 Matching Network	30
3.4.1.2 Network Analyser	32
3.5 CHAPTER SUMMARY	32

CHAPTER 4 SMARTBADGE DESIGN.....	33
4.1 CHIPCON CC1010	33
4.1.1 8051 Microcontroller	33
4.1.2 RF Transceiver	33
4.2 PERIPHERALS.....	35
4.3 FINAL DESIGN	36
4.4 CHAPTER SUMMARY	38
 PART II: SOFTWARE	39
CHAPTER 5 BADGE TO BADGE PROTOCOL	41
5.1 DATA FORMAT	41
5.2 INFRARED HARDWARE CONSIDERATIONS	42
5.3 BADGE-TO-BADGE SOFTWARE	43
5.4 CHAPTER SUMMARY	45
 CHAPTER 6 BADGE TO BASE PROTOCOL	47
6.1 ROUTING PROTOCOLS	47
6.1.1 Simulations	48
6.1.2 Results.....	49
6.1.2.1 Network Lifetime	49
6.1.2.2 Message Throughput.....	52
6.1.3 Analysis.....	53
6.2 MEDIUM ACCESS CONTROL PROTOCOLS.....	54
6.2.1 Network and Traffic Models.....	54
6.2.2 Uplink MAC (U-MAC) Protocol.....	55
6.2.3 Comparison to Other Protocols.....	57
6.3 U-MAC SIMULATIONS.....	58
6.3.1 OPNET Simulation Model.....	58
6.3.1.1 SmartBadge Nodes.....	61
6.3.1.2 Base Station Nodes	62
6.3.1.3 Central Server Node.....	63
6.3.2 Simulation Results	64
6.3.2.1 SmartBadge Results	64
6.3.2.2 Base Station Results	67
6.3.2.3 Central Server Results.....	71
6.4 CHAPTER SUMMARY	71
 CHAPTER 7 CONCLUSIONS	73
7.1 THESIS SUMMARY	73
7.1.1 Hardware.....	73
7.1.2 Software	73
7.2 FUTURE RESEARCH	74
7.2.1 Hardware.....	74
7.2.1.1 Battery Issues	74
7.2.1.2 Chipcon CC2431	75
7.2.1.3 Packaging	75
7.2.2 Software	75
7.2.2.1 Applications	76
 GLOSSARY.....	77

APPENDIX A	HARDWARE DETAILS.....	81
A.1	PERIPHERALS.....	81
A.1.1	<i>Infrared (IR) Transceiver</i>	81
A.1.2	<i>Liquid Crystal Display (LCD)</i>	82
A.1.3	<i>Piezoelectric Buzzer</i>	82
A.1.4	<i>Light Emitting Diodes (LEDs)</i>	83
A.1.5	<i>Pushbuttons</i>	83
A.2	SCHEMATIC.....	84
A.3	PCB LAYOUT.....	84
APPENDIX B	SMARTBADGE PROTOTYPES.....	87
B.1	FIRST PROTOTYPE.....	87
B.2	SECOND PROTOTYPE.....	87
B.3	FINAL DESIGN.....	89
APPENDIX C	ROUTING PROTOCOLS	91
C.1	DESTINATION SEQUENCED DISTANCE VECTOR ROUTING (DSDV).....	91
C.2	AD-HOC ON-DEMAND DISTANCE VECTOR ROUTING (AODV).....	91
C.3	NS-2 SIMULATION SCRIPT.....	92
APPENDIX D	OPNET SOURCE CODE.....	97
D.1	SMARTBADGE.....	97
D.1.1	<i>INIT Enter Executives</i>	97
D.1.2	<i>INIT Exit Executives</i>	99
D.1.3	<i>SLEEP Enter Executives</i>	99
D.1.4	<i>SLEEP Exit Executives</i>	99
D.1.5	<i>RX_CTS Exit Executives</i>	99
D.1.6	<i>RX_ACK Enter Executives</i>	99
D.1.7	<i>RX_ACK Exit Executives</i>	99
D.1.8	<i>Header Block</i>	99
D.1.9	<i>Function Block</i>	101
D.2	BASE STATION.....	107
D.2.1	<i>INIT Enter Executives</i>	107
D.2.2	<i>FORWARD Exit Executives</i>	108
D.2.3	<i>Header Block</i>	110
D.2.4	<i>Function Block</i>	111
D.2.5	<i>Temporary Variables</i>	111
D.3	CSMA.....	111
D.3.1	<i>idle Enter Executives</i>	111
D.3.2	<i>wt_free Enter Executives</i>	111
D.3.3	<i>tx_pkt Enter Executives</i>	111
D.3.4	<i>Header Block</i>	112
D.4	SERVER.....	112
D.4.1	<i>INIT Enter Executives</i>	112
D.4.2	<i>REPLY Exit Executives</i>	113
D.4.3	<i>Header Block</i>	114
D.4.4	<i>Temporary Variables</i>	114
D.5	RADIO PIPELINE.....	115
APPENDIX E	BASE STATION COVERAGE AREA DERIVATIONS	117
E.1	SCENARIO 1.....	117
E.2	SCENARIOS 2 AND 5.....	117
E.3	SCENARIOS 3 AND 6.....	118
E.4	SCENARIOS 4 AND 7.....	119
REFERENCES.....	121	

List of Figures

Figure 2.1 PCB layout of the MIT Thinking Tag [5]	6
Figure 2.2 The MIT Meme Tag [6]	6
Figure 2.3 The nTag has a large LCD display and three control buttons [7].....	7
Figure 2.4 CharmBadge fully packaged, and an earlier version given to the HITLab [2]	8
Figure 2.5 Lancaster University's Pendle [11]	8
Figure 2.6 Timing relationship between receiver and different senders, CS is Carrier Sense [15].....	13
Figure 2.7 The hidden terminal problem, the circles show transmission ranges of each node.....	13
Figure 2.8 The S-MAC and T-MAC duty cycles and their effect on a normal data flow [17]	15
Figure 2.9 The early sleeping problem [17].....	16
Figure 2.10 The FRTS message exchange [17]	16
Figure 2.11 The MS-MAC active zone (shaded) is created to quicken connection setup for mobile sensors [20].....	17
Figure 3.1 Yagi Antenna; feed on 2 nd element, 2cm element spacing,.....	25
Figure 3.2 Mesh Plate Antenna; total area 5cm x 5cm	25
Figure 3.3 Rectangular Loop Antenna; dimensions 5cm x 3cm.....	25
Figure 3.4 Circular Loop Antenna; 3cm radius	25
Figure 3.5 Double Quad "Bow Tie" Antenna; 4.5cm element lengths	26
Figure 3.6 Rectangular Spiral Antenna; feed through PCB in top left corner, outer dimensions 5cm x 3cm, 0.5cm spacing between spirals, total length 34.8cm ($\sim\lambda$ @ 868 MHz)	26
Figure 3.7 Yagi Antenna; gain at 0°: 8.6 dBi, gain at 60°: -3.5 dBi	27
Figure 3.8 Mesh Plate Antenna; gain at 0°: 6.5 dBi, gain at 60°: 3.0 dBi.....	27
Figure 3.9 Rectangular Loop Antenna; gain at 0°: 7.9 dBi, gain at 60°: 2.8 dBi.....	27
Figure 3.10 Circular Loop Antenna; gain at 0°: 8.1 dBi, gain at 60°: 0.0 dBi.....	28
Figure 3.11 Double Quad "Bow Tie" Antenna; gain at 0°: 8.9 dBi, gain at 60°: 0.5 dBi.....	28
Figure 3.12 Rectangular Spiral Antenna; gain at 0°: 6.7 dBi, gain at 60°: -1.0 dBi ...	28
Figure 3.13 Rectangular Spiral Antenna without ground; gain at 0°: 7.8 dBi, gain at 60°: 6.8 dBi	29
Figure 3.14 3D Radiation Pattern for Rectangular Spiral Antenna	30
Figure 3.15 Two 2-element matching networks are used to create the 3-element π network	31
Figure 3.16 (a)Impedance Matching Network (b)Component Values at 868 MHz	31
Figure 4.1 CC1010 Block Diagram [32].....	34
Figure 4.2 Block Diagram for RF Transceiver [32]	34
Figure 4.3 Manchester Coding.....	35
Figure 4.4 SmartBadge Block Diagram.....	37
Figure 4.5 The final SmartBadge design, front and back	37

Figure 5.1 UART data format of the letter ‘U’	42
Figure 5.2 Infrared data format of the letter ‘U’	42
Figure 5.3 SmartBadge partner search timing	43
Figure 5.4 Badge-to-Badge Protocol State Diagram	44
Figure 6.1 10 x 10 metre room, 20 nodes (green), range 5m (black ring is range of base station)	49
Figure 6.2 Network Lifetime with a single node transmitting	50
Figure 6.3 Network Lifetime with all nodes transmitting	50
Figure 6.4 Node Lifetime with LEDs removed and a single transmitter	51
Figure 6.5 Node Lifetime with LEDs removed and all nodes transmitting	52
Figure 6.6 Packet Delivery Rate with a single transmitting node	52
Figure 6.7 Packet Delivery Rate when every node is transmitting	53
Figure 6.8 U-MAC Data Packet Format	55
Figure 6.9 U-MAC Control Packet Format	55
Figure 6.10 U-MAC State Diagram from OPNET [45]	56
Figure 6.11 The network model for scenario 7, from OPNET [45]	59
Figure 6.12 The SmartBadge, base station and server objects of the network model	59
Figure 6.13 Base Station Coverage Areas	60
Figure 6.14 SmartBadge Node Model	61
Figure 6.15 U-MAC Retransmit Packet Format	62
Figure 6.16 Base Station Node Model	62
Figure 6.17 Base Station Process Model	63
Figure 6.18 Central Server Node Model	64
Figure 6.19 Central Server Process Model	64
Figure 6.20 SmartBadge Lifetime	65
Figure 6.21 Packets Sent/Received by the SmartBadges	66
Figure 6.22 End-to-End Delay	66
Figure 6.23 RTS Packets received by Base Stations	67
Figure 6.24 Data Packets received by Base Stations	68
Figure 6.25 Data Timeouts for each Base Station	68
Figure 6.26 Bit Error Rate per Base Station	69
Figure 6.27 Signal-to-Noise Ratio per Base Station	70
Figure 6.28 Channel Utilisation per Base Station	70
Figure 6.29 RTS, Data and Cancel packets at the Central Server	71
Figure A.1 Infrared Circuit Diagram	81
Figure A.2 LCD Circuit Diagram	82
Figure A.3 Buzzer Circuit Diagram	83
Figure A.4 LED Circuit Diagram	83
Figure A.5 Button Circuit Diagram	83
Figure A.6 SmartBadge Schematic	85
Figure A.7 PCB Layout Top Side	86
Figure A.8 PCB Layout Bottom Side	86
Figure B.1 The first prototype, front and back	87
Figure B.2 The second prototype design, front and back	88
Figure B.3 The extension to the second prototype necessary for the infrared transceiver	89
Figure B.4 The final SmartBadge design, front and back	89

Figure D.1 CSMA Process Model	112
Figure E.1 Coverage area for scenario 1 (shaded).....	117
Figure E.2 Coverage area for scenarios 2 and 5 (shaded).....	117
Figure E.3 Coverage area for scenarios 3 and 6 (shaded).....	118
Figure E.4 Drawings for the uncovered corner and half uncovered edge	118
Figure E.5 Coverage area for scenarios 4 and 7 (shaded).....	119

List of Tables

Table 2.1 Badge Features Comparison	9
Table 6.1 Room sizes and node counts for routing scenarios.....	48
Table 6.2 Power consumption breakdown for routing simulations	49
Table 6.3 OPNET simulation scenarios.....	60
Table 6.4 Time SmartBadges spend Sleeping, Transmitting, Receiving	65
Table 6.5 SmartBadge Retransmit (RTX) Requests	67

Chapter 1 Introduction

This thesis describes the hardware and software development carried out to create the SmartBadge as a project for the HITLabNZ. The SmartBadge is designed to be worn at a conference instead of a paper name tag as it uses electronics to enhance interactions between two participants (for example it can be used to automatically collect business card information). This introductory chapter begins by providing the motivation for the work including why it is useful. Then it presents the goals of the SmartBadge design and the contributions made by this thesis, and finally the structure of the rest of the thesis.

1.1 Motivation

A conventional name tag purely displays the wearer's name. An intelligent badge such as the SmartBadge however can enhance social interactions by also providing a greeting to the viewer or show a connection between the two people (e.g. matching interests) or record people a wearer has met.

The SmartBadge is primarily intended for use at conferences and other large gatherings. The intended application is as a replacement for both business cards and name tags. It will collect not only another person's contact details but also log how long you spend talking to them so that the contacts can be organised by conversation length making it easier to find people that were of interest to you. The SmartBadges could also use this information to direct you to another person with similar interests and hence could be used to enhance networking opportunities at these events. Other possible applications are described in section 7.2.2.1.

The main advantage of having an electronic badge like the SmartBadge instead of just a standard paper badge is that it will seamlessly collect information from people you meet so you do not have to remember to do so. It can also be used to greet a new contact or to show interest matches between users both of which can enhance social networking by acting as an icebreaker.

1.2 Design Goals

The SmartBadge idea came from the CharmBadge [2] which is a product of Charmed Technology, one of the HITLabNZ's consortium members. In order to create an electronic badge that will enhance the conference experience the following design goals should be addressed:

- Radio Frequency communication in addition to Infrared. The main feature to be added was to supplement the infrared transceiver that the CharmBadge uses with a radio link to be used for real time data collection.
- Improve the user interface. The CharmBadge only has four LEDs and two buttons so in order to be used as a name tag it sits in a lanyard along with a piece of paper with your name on it and this is worn around the neck. Our goal was to add an LCD display for your name which can then be used for other information. We also kept the LEDs and buttons and added a buzzer for use in possible future applications.
- Small and Lightweight. Since this is a badge to be worn on a shirt pocket these were important physical criteria. The goal was to have the final design roughly the size of a credit card (86 x 54 mm) and to keep the weight down as much as possible. This meant using surface mount components and small batteries.
- Power Conservation. As it runs on batteries and should last at least for one whole day investigating low power, energy efficient designs was also important and this is reflected in the communication software.
- Cost. In order to justify supplying a SmartBadge to every conference participant the cost must be kept as low as possible.

1.3 Thesis Contributions

This thesis is divided into two parts, namely the hardware and software design, and it makes contributions in both of these areas.

For the hardware the thesis designs a badge with two different communication links (infrared and radio) and uses these to collect information in real time. This could be an important feature for some applications and also reduces the workload of event organisers after the event has ended as all the data has already been collected. The SmartBadge we have designed also has a better user interface than other badges as it has more peripherals to interact with the user. These can provide a better experience

for the user and also can enable a larger range of applications. These advantages of the SmartBadge over existing intelligent badges are summarised in Table 2.1. A major contribution to the hardware design has been the antenna for the RF transceiver, as described in Chapter 3.

The major original contribution within the software is the development of a new Medium Access Control (MAC) protocol (Uplink MAC, or U-MAC) specifically tailored to the SmartBadge network and any other networks whose main goal is to send collected data back to a central server for processing. This is important as current low power protocols are primarily focussed on networks that involve a lot of communication among all nodes, rather than all talking to a central node. By designing a protocol tailored to the latter scenario further power can be saved.

1.4 Thesis Structure

Chapter 2 describes previous work in the area. It describes other intelligent badges that have been made and compares their features. It also gives details of some recent MAC protocols as background work for the development of U-MAC.

The remainder of the thesis is then split into two parts. The first part deals with the hardware design. It begins with Chapter 3; a description of the design of the antenna for the RF section. This is followed by an overview of the CC1010 microcontroller and RF transceiver IC and the rest of the SmartBadge design.

Part II describes the software developed for the SmartBadge, in particular that for the communication between two SmartBadges and between a SmartBadge and a base station. First we describe the badge-to-badge protocol which identifies what happens with the infrared transceiver. Then Chapter 6 details the badge-to-base protocol design and simulations. It first mentions an approach based on routing protocols but then shifts focus to MAC protocols for reasons discussed there.

Finally conclusions are presented where we discuss what needs to be done to the SmartBadge hardware and software before it can be mass produced. We also present some possible applications for the SmartBadge. This is followed by a glossary of terms used and the appendices.

Chapter 2 Previous Work

Previous work relevant to the SmartBadge is presented here. Firstly for the hardware we provide some background on previous intelligent badges and compare their features to those of the SmartBadge. Note that the SmartBadge is the only one to combine both infrared and radio communications and also has a wider array of peripherals than existing badges. Then we discuss why Bluetooth [3] and RFID [4] are not suitable for the SmartBadge network design. We also describe some existing Medium Access Control (MAC) protocols that have been developed for sensor networks (and hence are power aware) as the major contribution of our software development is the design of a new power conserving MAC protocol. This new protocol has been tailored to the SmartBadge network and so is more energy efficient than the existing protocols as shown in section 6.3.

2.1 *Intelligent Badges*

There are a number of other intelligent badges but none of them implement the same features as the SmartBadge. We describe a few of these starting with the Thinking Tag [5], Meme Tag [6] and nTag [7] from the MIT Media Lab's [8] work. Then we describe the CharmBadge [2], which was the basis for the SmartBadge, and also some others of interest.

2.1.1 Thinking Tag, Meme Tag and nTag

The first intelligent badges are due to work from Rick Borovoy and the MIT (Massachusetts Institute of Technology) Media Lab [8] in 1995; these were called Thinking Tags [5] (see Figure 2.1). They communicated via infrared and had five LEDs on them which would light up when the person you were talking to had chosen the same answer to some multi-choice questions. This was done during an open evening at the Media Lab, where the users had to get their answers programmed into the badge when they arrived by dunking them into a programming bucket. The event was a success as the users felt that the tags did enhance their social interactions without being too intrusive, the buckets also proved a popular and simple method of programming the Thinking Tags.

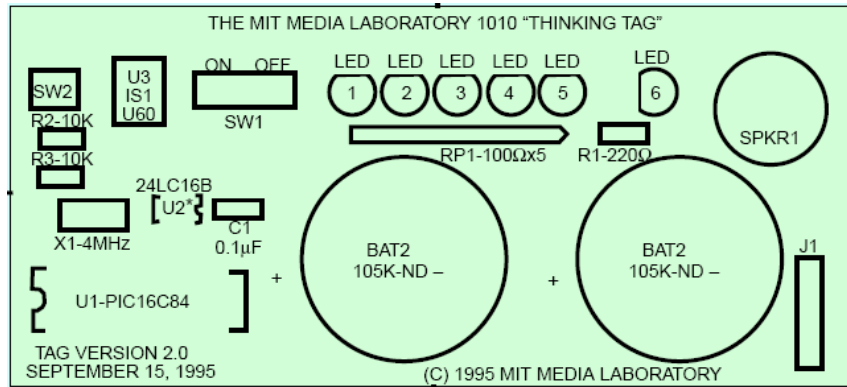


Figure 2.1 PCB layout of the MIT Thinking Tag [5]

These were followed in 1997 with the so-called Meme Tags [6] pictured in Figure 2.2. They used infrared to communicate and exchange so-called memes (or short phrases) between badges. When two badges made contact they would each display a unique meme on their LCD screens which could then be traded by using the ‘accept’ or ‘reject’ buttons. This movement of memes amongst a group was displayed during the event by downloading collected memes from a badge while it was connected to a computer which was also used to create new memes. Users found these badges more intrusive and took a while to get used to the idea that the display on their badge was for the other person, not themselves as in pagers or cell phones. However the Meme Tags did create a sense of being the centre of attention as all the tags one saw had their own name and this also caused a focused interaction between users as they shared in an exclusive meme exchange.

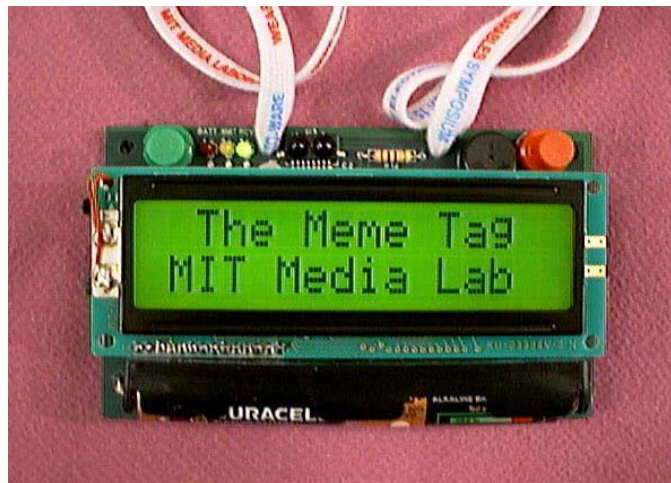


Figure 2.2 The MIT Meme Tag [6]

Rick Borovoy has since set up a company which markets more advanced tags known as nTAGs [7] that are illustrated in Figure 2.3. These tags are still infrared based but

have a much larger LCD display and run on four 'AAA' batteries. They incorporate the functions of the previous badges as well as many others listed on the website [9], such as being able to administer polls or surveys electronically or to be able to use the nTAGs ID to personalise message boards so that a user is more easily aware of an awaiting message.



Figure 2.3 The nTag has a large LCD display and three control buttons [7]

2.1.2 CharmBadge

Charmed Technology is another spin-off company from the MIT Media Lab, and is also a HITLabNZ consortium member. The CharmBadge [2] is hence the badge that our design is derived from as the HITLab had a couple of samples (see Figure 2.4). It is about the size of a credit card and again communicates via infrared. As well as keeping track of the contact time between two badges (called dwell time) it also displays an affinity score on four LEDs similar to the Thinking Tags; the more pre-set questions answered the same between two badges the more LEDs light up. The website [2] describes the badges as follows:

“Each CharmBadge is assigned an IPv6 address, which corresponds to the user's contact information in the CharmBadge database. The CharmBadge exchanges this IPv6 address as well as its affinity information with other badges in its field of view via infrared transmit/receivers and tracks the time it sees another CharmBadge as an interaction. In order to access the actual contact information, the badge is downloaded at a CharmBadge Download Center and the interactions transferred to the CharmBadge database. The user may then visit the CharmBadge Information Access Center to view their contacts corresponding to that event.”

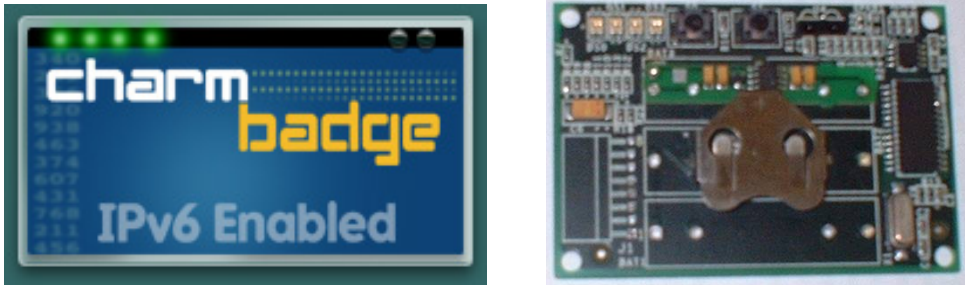


Figure 2.4 CharmBadge fully packaged, and an earlier version given to the HITLab [2]

2.1.3 Pendle

Lancaster University's Computing Department [10] created a badge they call a Pendle [11] (Figure 2.5) in 2003. It uses RF to talk to nearby computers and other displays, to adapt the displayed information to a set of user preferences (keywords and URLs) stored on the badge. It also has touch and acceleration sensors which allow the user to interact with the Pendle by tilting it in certain directions. The Pendle is in an implicit state when worn where the user's keywords are used to create the personalised displays. When a user grabs the badge the touch sensor recognises this and the Pendle enters an explicit state where holding the Pendle displays the next URL in the list, and shaking it (detected by the accelerometer) removes the displayed URLs.

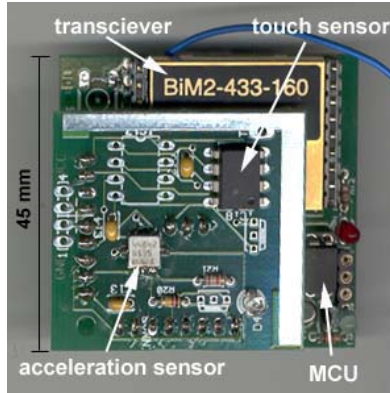


Figure 2.5 Lancaster University's Pendle [11]

2.1.4 RFID Badges

There are many other badges available designed to track people or products as they move around a building. These involve having readers located throughout an office so that a user's location is known and this can be used for example to redirect an incoming phone call to the phone nearest a user's current location, rather than just to their office phone, should they be on the move. Some companies selling them include RF Technologies [12] and Access Inc [13] for RFID based badges and Varitronics Inc [14] which has an infrared based solution.

2.1.5 Comparison of Badge Features

Table 2.1 summarises the features of the badges described above and also the intended features of the SmartBadge. It can be seen from this table that the SmartBadge is the only one capable of collecting data in real time (except for the nTag which only works when users are in the right location), which is accomplished over the radio link. It also has more peripherals than any of the others (LCD, LEDs, buzzer and buttons).

In addition to improving on previous electronic badge hardware, the SmartBadge also has a better communication protocol (as described in Chapter 6) and sufficient memory space for more complex applications, examples of which are described in section 7.2.2.1. Specifically the real time data collection (a unique SmartBadge feature) could be useful in applications like conducting polls or awarding prizes for the first person to complete an icebreaker exercise.

Feature	Thinking Tag	Meme Tag	nTag	CharmBadge	Pendle	SmartBadge
Infrared	Y	Y	Y	Y	-	Y
Radio Link	-	-	*	-	Y	Y
Batteries	2 x CR2032	2 x AA	4 x AAA	1 x CR2032	1 x CR2032	1 x CR2032
LEDs	5	-	-	4	-	4
LCD	-	2 x 16 characters	128 x 64 pixels	-	-	2 x 16 characters
Buttons	-	2	3	2	-	2
Buzzer	Y	Y	-	-	-	Y
Accelerometer	-	-	-	-	Y	-
IPv6 Enabled	-	-	-	Y	-	-
Real Time Data Collection	-	-	*	-	-	Y
Microcontroller	PIC16C84	MC68HC11-F1FN	†	PIC16F876	PIC16F876	CC1010
Memory Space	1,892B	13,824B	†	14,960B	14,960B	34,944B

Table 2.1 Badge Features Comparison

* The nTag uses RFID to collect data during an event but this can only be done when users pass close to an RFID reader e.g. when passing through a doorway

† As the nTag is a commercial product such details are unavailable

2.2 CC1010 versus Other Technologies

The SmartBadge is centred around the Chipcon CC1010 microcontroller and RF transceiver (see section 4.1), but when we describe the SmartBadge system to others there are two other RF technologies that often get mentioned as possible alternatives. These are Bluetooth® and RFID (Radio Frequency Identification). This section presents a few brief reasons specific to each technology which shows why they are not suitable for the intended SmartBadge network.

2.2.1 CC1010 versus Bluetooth®

The first alternative to using the CC1010 radio IC would be to buy a Bluetooth® chip which already implements both the link layer and the application layer. A very good reason for not doing so is that the network architecture of Bluetooth® is not ideal for the SmartBadge network which should be capable of handling at least a few hundred SmartBadges at the same event.

The Bluetooth® website [3] states:

“The Bluetooth wireless technology is designed to replace cables between cell phones, laptops, and other computing and communication devices within a 10-meter range.”

This points out that Bluetooth® has not been designed to network large numbers of devices together. Rather it is for creating Personal Area Networks (PANs) to allow consumer devices to communicate wirelessly.

Further, Bluetooth® uses a master / slave architecture where one device becomes the master and initiates all communication with slave nodes. The Bluetooth® piconet architecture does not allow a large number of simultaneous connections. It allows for up to only 7 slaves for each master meaning that we would need to create multiple piconets and hence multiple masters which would require extra processing (and hence reduced lifetime) on all these master nodes.

In addition the Bluetooth® network uses a frequency hopping scheme and so tight clock synchronisation between master and slave nodes is required. This would create a lot of control overhead considering we will not need to send a vast amount of data. It would also require a new protocol suite to be written.

Finally, Bluetooth® is not very well suited to extended operation using small capacity batteries, as intended for the SmartBadge, since its power consumption is too high.

2.2.2 CC1010 versus RFID

Another possible technology is RFID or radio frequency identification. RFID systems use either passive or active tags primarily to track products through a supply chain. A good source of information on RFID is the RFID Journal [4]. We would not be able to use passive tags for the SmartBadges as the data to be sent to the server is changing. Active tags still have a battery and those with a decent range (greater than 1 metre) operate at the same frequency as the CC1010, so the energy savings compared to the CC1010 approach would be negligible.

In order to read multiple SmartBadges within range of a single reader would require a TDMA (Time Division Multiple Access) scheme to be established by the reader and hence the SmartBadges would have to be carefully synchronised so that they are awake at the right time (or else always stay awake which uses more battery power). Our system using the CC1010 on the SmartBadge as the master (i.e. device starting the data transfer) eliminates the need for synchronisation as the base stations are always on, running off mains power, so power conservation is not an issue for them.

The nTag system does use RFID but points out that this connection is only active when a user passes through a doorway presumably so that the range is small enough to eliminate the problem of reading multiple tags at once, however we want our system to be able to upload data at regular intervals, not just when the user is standing in the right position.

2.3 MAC Protocols

Electronic badges need a good communications protocol in order to effectively exchange data. Our solution to this network design problem was to create a novel MAC (Medium Access Control) protocol and so we provide here an overview of some existing MAC protocols for wireless sensor networks.

MAC protocols are necessary as they provide a means of deciding which device has the right to use the medium (radio link in this case) at any particular point in time. This is done to avoid all the devices sending data simultaneously which would cause collisions, and hence no data would actually get through the channel.

2.3.1 Sensor MAC (S-MAC)

There have been a number of MAC protocols for wireless ad-hoc sensor networks created in the past few years, most of which are variations of the Sensor MAC (S-MAC) protocol developed by Wei Ye [15].

The primary goal in creating S-MAC was to reduce energy consumption as it is designed for sensor networks which need to survive a long time on a single set of batteries. The protocol achieves good scalability and collision avoidance through utilising a combined scheduling and contention scheme.

Ye [15] identifies four main sources of energy waste and develops the protocol to minimise these. These are:

- collisions,
- overhearing,
- control packet overhead, and
- idle listening.

Idle listening is the major energy user since in sensor networks the nodes are often doing nothing until an event occurs. Thus the main idea is to put nodes to sleep as often as possible rather than going into an idle listening mode as in conventional protocols. The other major components of S-MAC are collision and overhearing avoidance and message passing.

The basic sleeping scheme means that a node goes to sleep for some time (i.e. turns its radio off) then wakes up to listen for messages destined for it. All nodes can choose their own sleep/wakeup cycle however neighbouring nodes should mutually synchronise where possible. Schedules are broadcast in a SYNC packet so that all immediate neighbours can talk to each other even if it means an individual node is following multiple schedules. This means that nodes need to be able to receive both SYNC and data packets so the listen interval is divided in two as shown in Figure 2.6. In this figure sender 1 only sends the SYNC packet, sender 2 only data and sender 3 sends both a SYNC packet and an RTS (Request to Send) which is required to reserve the medium before sending data.

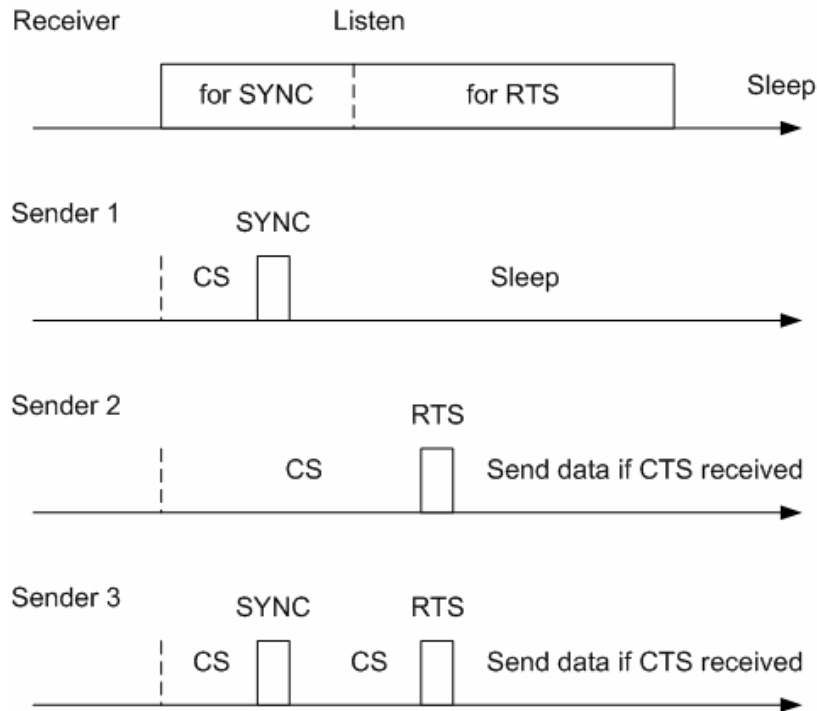


Figure 2.6 Timing relationship between receiver and different senders, CS is Carrier Sense [15]

S-MAC is a contention based protocol and hence needs to implement a collision avoidance scheme. The scheme implements RTS/CTS packets to counter the hidden terminal problem, similar to the IEEE 802.11 [16] scheme. The hidden terminal problem occurs when there is a series of nodes just within range of each other as shown in Figure 2.7. The problem is that if A is sending data to B then C cannot hear this and may send out its own data to D which would also interfere at B. To avoid this A sends a Request to Send (RTS) packet to B who replies with a Clear to Send (CTS) packet so that nodes that can only hear either A or B know to keep quiet when they overhear an RTS or CTS to avoid interfering with A's data.

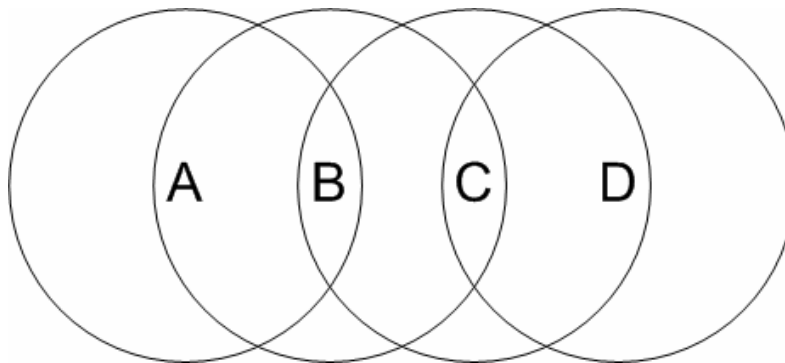


Figure 2.7 The hidden terminal problem, the circles show transmission ranges of each node

S-MAC also includes a virtual carrier sense by sending a duration field with each packet so that nodes that hear this packet know they will have to wait at least this amount of time before trying again (and so can sleep meanwhile). To avoid overhearing nodes will also go to sleep when they hear an RTS or CTS packet not addressed to them as transmitting during this time will interfere with another nodes transmission.

The other feature of S-MAC is its message passing. It sends long messages as a burst of smaller packets and reserves the medium for all the fragments at once. This approach means that the receiver will get the full message quickly as it is likely to not be able to process it until all the data arrives. However the approach does not promote node level fairness as other nodes may have to wait a long time to send anything. The justification for this is that in a typical sensor network all nodes collaborate on a common application and hence it is only application level fairness that really matters.

2.3.2 S-MAC Based Protocols

The main disadvantage of the S-MAC protocol is that the energy savings are a trade off with throughput. Since a node is sleeping when idle any data to be sent to it has to wait until the receiving node wakes up again. The protocols mentioned below all suggest improvements to S-MAC by addressing this problem.

2.3.2.1 Timeout MAC (T-MAC)

Timeout MAC [17] is designed to handle traffic variations by dynamically ending the active part of the sleep/listen cycle. It claims to achieve similar energy savings to S-MAC under a constant load (both consume up to 98% less energy than Carrier Sense Multiple Access or CSMA) but to outperform S-MAC under a variable load by a factor of 5 as S-MAC has a fixed duty cycle (it can be adjusted a priori to suit the application but otherwise remains constant).

T-MAC works by entering a sleep state as soon as no activation event has occurred for a time T_A as shown in Figure 2.8. This means that messages are sent in bursts at the start of the active time and so the active time is likely to be shorter in most cycles than the fixed active time of S-MAC (also in Figure 2.8). Note that T_A has to be at least as long as the total of the (fixed) contention interval plus the length of an RTS packet plus the delay between receiving an RTS and sending a CTS packet.

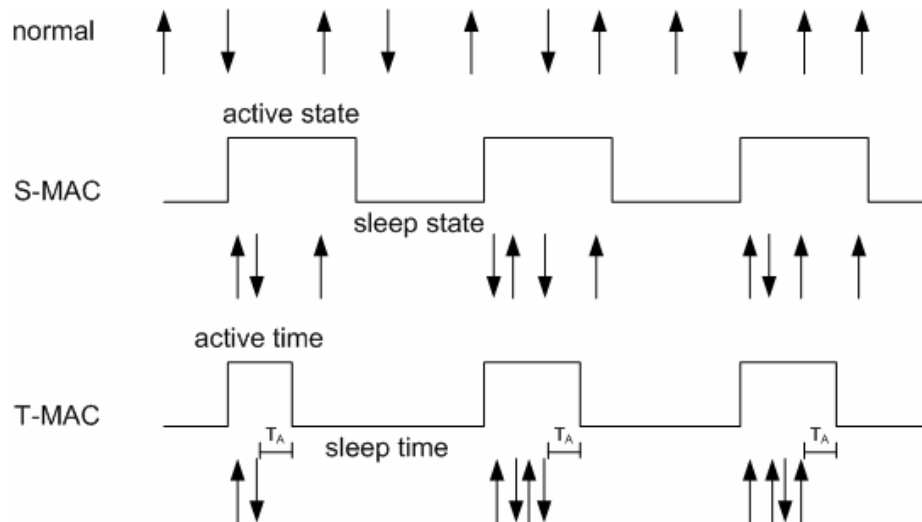


Figure 2.8 The S-MAC and T-MAC duty cycles and their effect on a normal data flow [17]

The activation event is defined as one of the following:

- the firing of a periodic frame timer;
- the reception of any data on the radio;
- the sensing of communication (via an RSSI level) on the radio, e.g. during a collision;
- the end-of-transmission of a node's own data packet or acknowledgement;
- the knowledge, through overhearing prior RTS and CTS packets, that a data exchange of a neighbour has ended

T-MAC uses the same RTS/CTS/DATA/ACK format as S-MAC but when an RTS is sent and no CTS received it retries up to three times since the receiving node could be asleep but it could also have not heard the RTS due to collision or be unable to reply due to an overheard RTS or CTS from another node. In these cases the transmitting node would return to sleep even though the receiving node was actually still awake.

T-MAC also has a mechanism to avoid what is called the early sleeping problem shown in Figure 2.9. Messages are only sent A to B to C to D but in order for C to send a message it has to win the medium by not hearing an RTS or CTS from B. If C hears a CTS from B it can not send an RTS to D as that would interfere with the data from A to B and so D would go to sleep even though C has data to send for it.

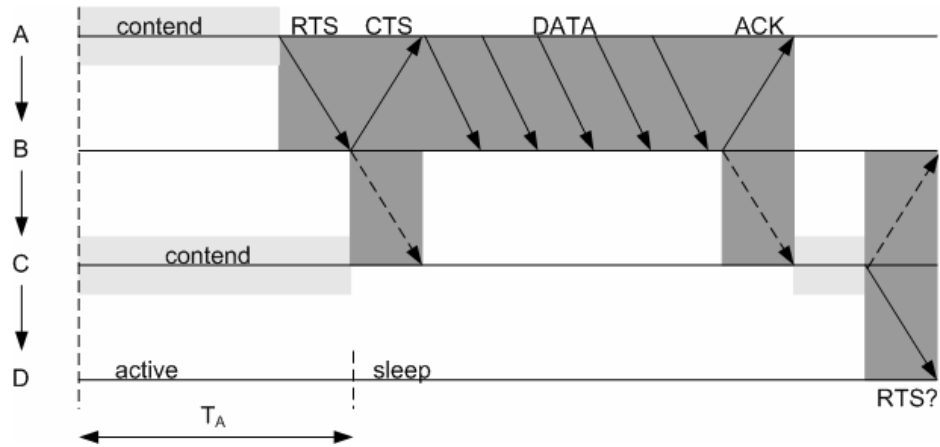


Figure 2.9 The early sleeping problem [17]

To solve this T-MAC creates a future-request-to-send (FRTS) packet which C sends on a CTS from B in order to keep D awake (see Figure 2.10). This means that before A sends data it must send a data-send (DS) packet which is the same size as a FRTS packet. The DS packet will collide with the FRTS at B but it contains no data so this is fine and it means that no data will be lost at B.

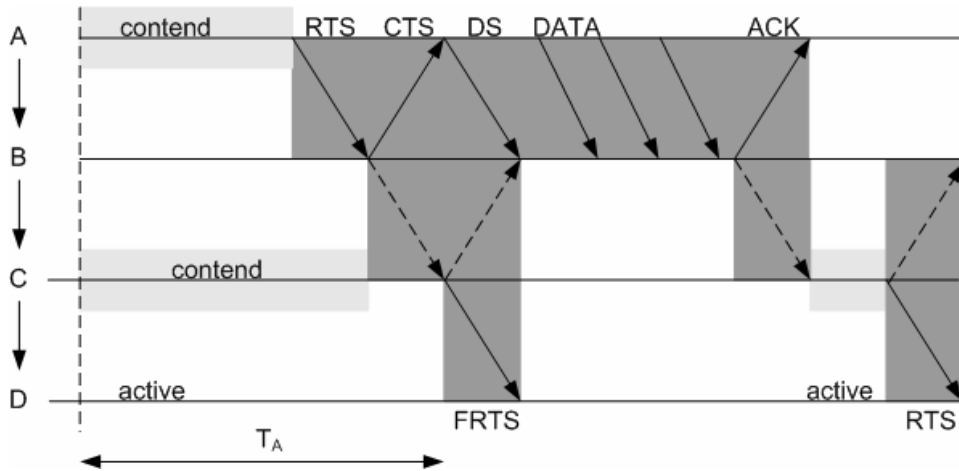


Figure 2.10 The FRTS message exchange [17]

The other solution proposed is full-buffer priority which means that when C's buffer is full it needs to send to D before it can receive from B so it will send out an RTS to D when it hears an RTS from B, rather than a CTS reply, thereby effectively winning the medium from B.

2.3.2.2 Wireless Sensor MAC (WiseMAC)

WiseMAC [18] is based on the preamble sampling technique developed by the same author [19]. This technique means the nodes sample the medium regularly, i.e. listen to the medium for the duration of a modulation symbol. If the medium is found busy

the node will listen until data is heard or the medium becomes idle. When transmitting, nodes employ a wake-up preamble of length equal to the sampling period to ensure the receiver is awake when data starts. The aim of WiseMAC is to reduce this preamble length. It does this by learning the sampling schedule of a node's direct neighbours such that the minimum preamble duration (T_p) is given by

$$T_p = \min(4\theta L, T_w) \quad (1)$$

where θ is the frequency tolerance of the clock crystal used, L is the interval (in seconds) between communications, and T_w is the sampling period (also in seconds). As a result of this WiseMAC is able to provide both low power and low hop delay while S-MAC and T-MAC can only provide one or the other. WiseMAC also achieves greater throughput than either S-MAC or T-MAC. For further details of WiseMAC see [18].

2.3.2.3 Mobility aware Sensor MAC (MS-MAC)

MS-MAC [20] uses Received Signal Strength Indicator (RSSI) measurements to determine the relative position of nodes and so can update a node's list of neighbours more quickly when the node moves. This gives better throughput than S-MAC in scenarios with mobile sensors. MS-MAC does this by creating an active zone around a mobile node and any border nodes as shown in Figure 2.11.

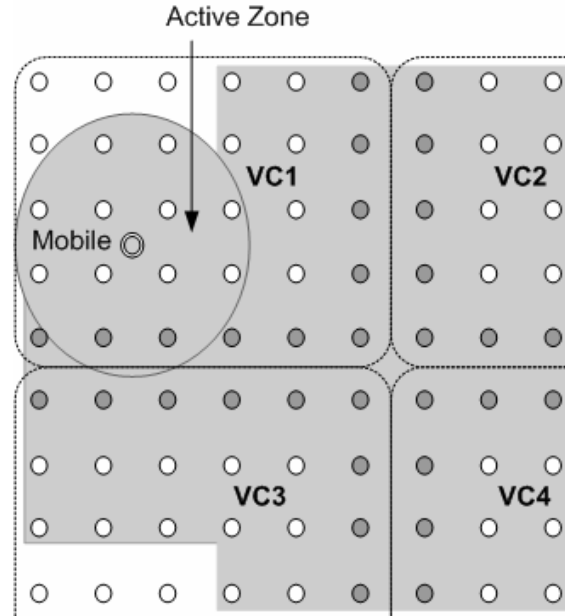


Figure 2.11 The MS-MAC active zone (shaded) is created to quicken connection setup for mobile sensors [20]

The active zone is an area where nodes will be active more frequently than in S-MAC so that mobile nodes can successfully migrate to the new virtual cluster's (VC) schedule¹. The active nodes immediately go into the synchronisation period (rather than waiting for the next scheduled one) and repeat it more often than they would in the default stationary case.

In order to create this active zone the MS-MAC SYNC messages also contain a hop count to the nearest mobile node and nearest cluster border nodes. In addition, if the mobile node is approaching another node (i.e. the hop count is decreasing) the node renews its active status by going straight to synchronisation again (and resetting its active status timer).

2.3.2.4 Other Protocols

There are several other MAC protocols which are also based on S-MAC. These include:

- Data-gathering MAC (DMAC) [21]. DMAC is optimised for unidirectional data gathering trees.
- CDMA Sensor MAC (CSMAC) [22]. CSMAC is a self organising location aware protocol that balances energy efficiency with latency, accuracy and fault tolerance.
- Dynamic Sensor MAC (DSMAC) [23]. DSMAC uses a dynamic duty cycle to trade off between energy efficiency and latency with minimal overhead.
- Pattern MAC (PMAC) [24]. PMAC has an adaptive sleep/wakeup cycle based on the traffic of a node and its neighbours. It saves more power than S-MAC for light loads and achieves higher throughput for heavy loads.

The above protocols are not ideal for the traffic and network models that the SmartBadges will operate under as described in section 6.2.1. DMAC is only useful for a multi-hop network. CSMAC is designed for the stringent latency requirements of a battlefield sensor network and the SmartBadge traffic will not require such extremely low latency. DSMAC and PMAC both have varying sleep times which are an extra complexity given the regular fixed message size of the SmartBadge traffic. For more details of the above protocols see the relevant papers.

¹ a virtual cluster (VC) is a collection of nodes following the same schedule

2.4 Chapter Summary

In this chapter we have described a number of other intelligent badges that have previously been developed. These include the Thinking Tag, Meme Tag and nTag which all came from the work of Rick Borovoy. Then there was the CharmBadge which was the basis for the SmartBadge. This was followed by the Pendle and some RFID based badges. This ended with a comparison of all the features of these badges against the SmartBadge (whose final hardware design is in Chapter 4) and a comparison of the CC1010s RF transceiver to both Bluetooth[®] and RFID. The SmartBadge improves on these existing badges by offering both infrared and radio communications and a greater number of peripherals which could be used to create richer applications.

The second half of this chapter looked at different MAC protocols. It began with Sensor MAC (S-MAC), which created the idea of sleep / wakeup cycles, and then covered others derived from S-MAC including T-MAC, WiseMAC and MS-MAC. T-MAC added a timeout to dynamically end the active period, WiseMAC reduced this active period by learning preamble lengths and MS-MAC was designed to handle mobility better than in S-MAC. Finally we briefly mentioned a number of other MAC protocols which have also been developed for sensor networks. The description of all these MAC protocols is necessary since we design a new MAC protocol in section 6.2.2

Part I:

Hardware

In the hardware section we first, in Chapter 3, look at the design of the antenna for the RF section including the goals of the antenna, simulations of various designs with SuperNEC [1], selection of a design for prototypes, testing and measurements of the impedance, and impedance matching for the chosen antenna.

Next Chapter 4 gives an overview of the Chipcon CC1010 microcontroller and RF transceiver as this IC is at the centre of the design. It also includes the rest of the hardware design for the SmartBadge. It mentions all the peripherals attached to the CC1010 to make up the SmartBadge and presents a block diagram of the overall design showing what signals are sent to each of the peripherals. Full details are provided in Appendix A.

Chapter 4 also presents the final constructed SmartBadge. This was a result of numerous prototypes which are discussed in Appendix B.

Chapter 3 Antenna Design

In this chapter we describe the antenna design for the RF section of the SmartBadge. It is divided into four parts. The first describes the goals for the antenna design, and the second discusses near field communications. Then there are details of the first antenna including simulations of different possible layouts and their results. The simulations have been carried out with SuperNEC [1] which is a hybrid Method of Moments (MoM) / Unified Theory of Diffraction (UTD) program for analysing the electromagnetic performance of antenna designs. Last is the second antenna design including calculations and testing of its impedance and the associated matching network between the antenna and the RF transceiver.

3.1 Design Goals

Antenna design is important in any radio communication system as the antenna controls the direction that the RF signal is sent (and received) from. It also provides a gain in the received signal hence meaning less power is needed to successfully communicate over a certain distance.

The main design goals for the antenna match those of the overall badge – to be lightweight, low profile and energy efficient. In order to achieve the first two it was decided to create an antenna from PCB tracks which would hence be flat and light. Energy efficiency meant looking for an antenna with high gain.

When we began the design there was going to be an RF link for both badge-to-badge and badge-to-base communication. Hence we needed an antenna design that was directional so that when communicating with other badges it would be possible to only interact with the person standing opposite you. These criteria lead us to the prototype design discussed in section 3.3.

Having already built the first prototype it was suggested that it may in fact be better to use infrared for badge-to-badge communications similar to the CharmBadge [2]. This had the advantage that the RF antenna could now be designed to be more omnidirectional which would make it easier to communicate with the base station. It also resulted in a design which took less space than the first. This is discussed in section 3.4.

3.2 Background

Through background reading we found a scheme called Near Field Communication (NFC) [25] [26] [27] which would be very low power as it is designed to work over short distances (up to 20cm). We had thought about looking for ways to extend the range just enough to be useful yet still minimise the likelihood of communicating with unintended badges but discovered that the near field region is not large enough given the chosen operating frequency of the CC1010.

The near field region is inside a sphere with a radius r given by [28]:

$$r \leq 0.62\sqrt{D^3/\lambda} \quad (2)$$

where λ is the wavelength and D the largest dimension of the antenna. Given that the operating frequency is 868 MHz and the largest dimension will be around 5cm this gives a radius of less than 1.2cm and hence ruled out any chance of using the near field with the given RF transceiver.

As a result this meant that the antenna designed for the SmartBadge would operate in the far field region and so could be analysed using the Method of Moments. We hence simulated the radiation patterns of different antenna designs using the SuperNEC [1] simulation software which runs on top of MATLAB [29].

3.3 Prototype Antenna

Using SuperNEC we tested some of the built in antenna structures and also tried a couple of others. The built in patterns used were the yagi, which is similar to most television antennas, a mesh plate, and rectangular and circular loops. The other patterns we tried were one based on a double quad bow tie [30] which is a homemade design intended to tap into IEEE 802.11 [16] hotspots, and a rectangular spiral design as this would be a more efficient use of space than a simple loop. These designs are shown in Figure 3.1 through Figure 3.6 respectively. They all assume a ground plane situated 1.6mm below the surface (this is the other side of the PCB in practice) and are set to operate at 868 MHz; the chosen CC1010 operating frequency. Other relevant information is specified in the figures.

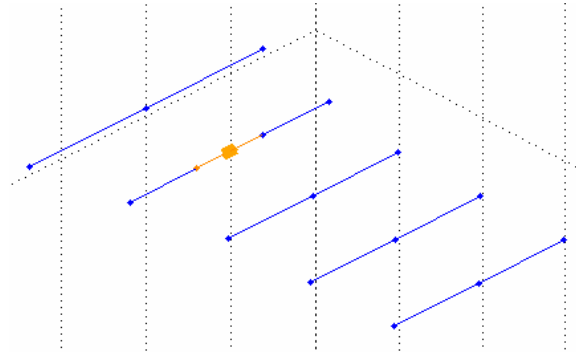


Figure 3.1 Yagi Antenna; feed on 2nd element, 2cm element spacing, element lengths 5.5cm, 4.7cm, 4cm, 4cm, 4cm respectively

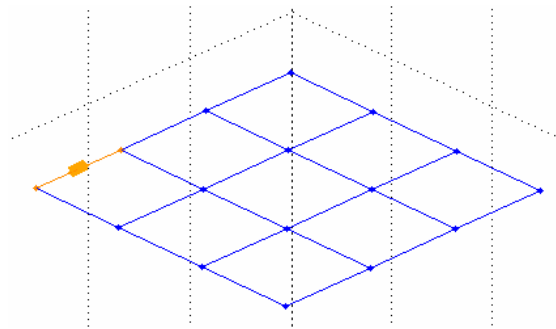


Figure 3.2 Mesh Plate Antenna; total area 5cm x 5cm

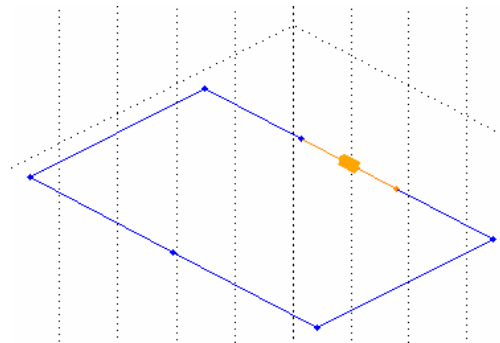


Figure 3.3 Rectangular Loop Antenna; dimensions 5cm x 3cm

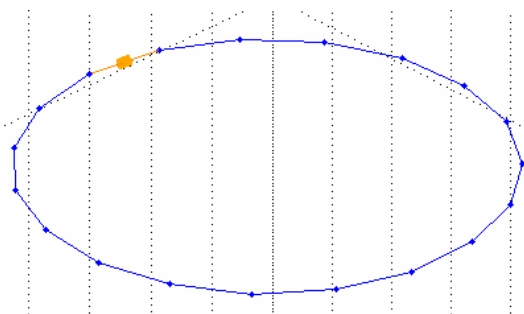


Figure 3.4 Circular Loop Antenna; 3cm radius

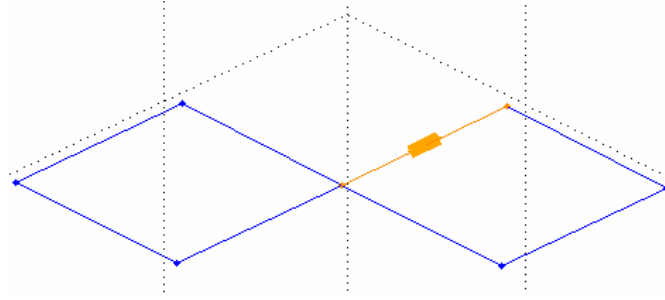


Figure 3.5 Double Quad “Bow Tie” Antenna; 4.5cm element lengths

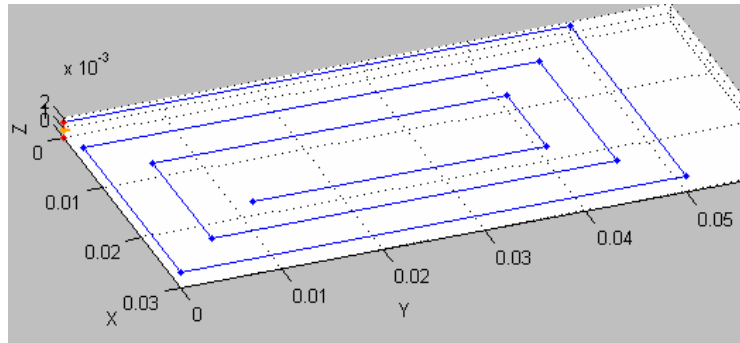


Figure 3.6 Rectangular Spiral Antenna; feed through PCB in top left corner, outer dimensions 5cm x 3cm, 0.5cm spacing between spirals, total length 34.8cm ($\sim\lambda$ @ 868 MHz)

3.3.1 Results

As we were looking for a directional design the results that were of most interest to us were the elevation radiation patterns² which show the antenna gain at different angles from the normal (i.e. z-axis) and hence can show which design has a smaller radiation beam spread. These results are in Figure 3.7 through Figure 3.12 respectively. The term dBi is the gain relative to an isotropic antenna (one that radiates evenly in all directions).

² These are all cross section cuts taken at $\phi = 88^\circ$ (i.e. a cross section almost along the y-axis)

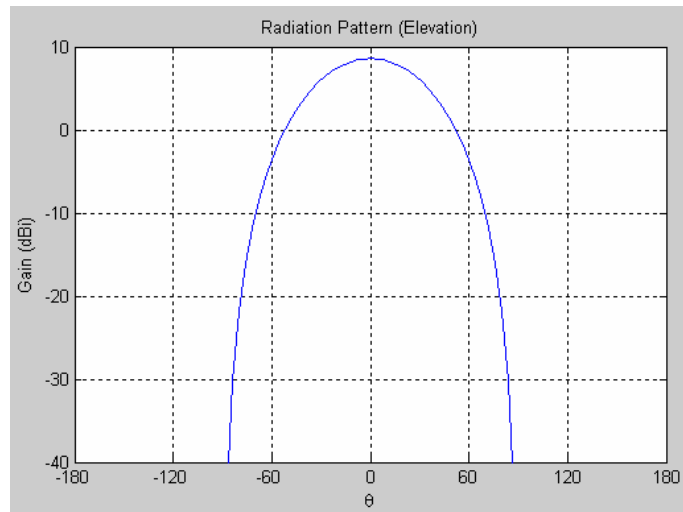


Figure 3.7 Yagi Antenna; gain at 0° : 8.6 dBi, gain at 60° : -3.5 dBi

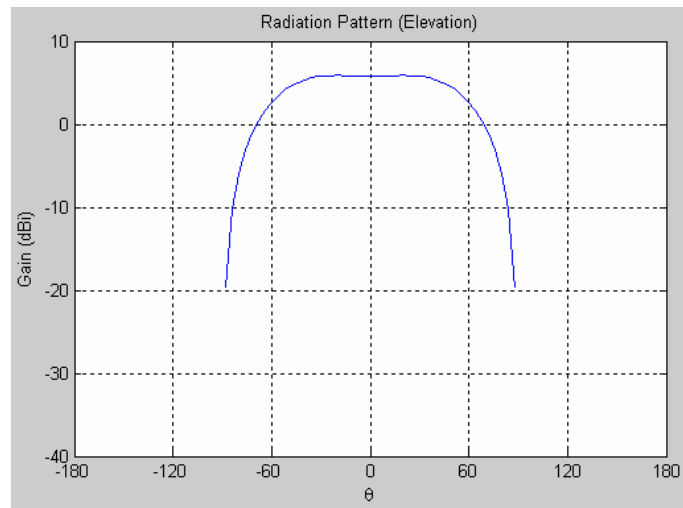


Figure 3.8 Mesh Plate Antenna; gain at 0° : 6.5 dBi, gain at 60° : 3.0 dBi

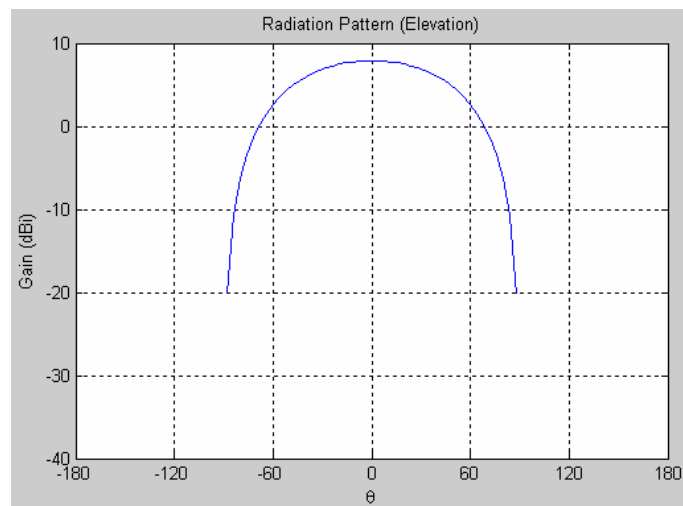


Figure 3.9 Rectangular Loop Antenna; gain at 0° : 7.9 dBi, gain at 60° : 2.8 dBi

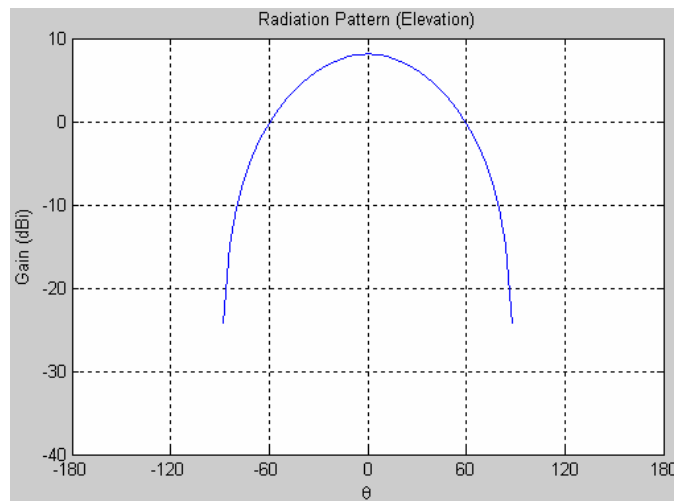


Figure 3.10 Circular Loop Antenna; gain at 0° : 8.1 dBi, gain at 60° : 0.0 dBi

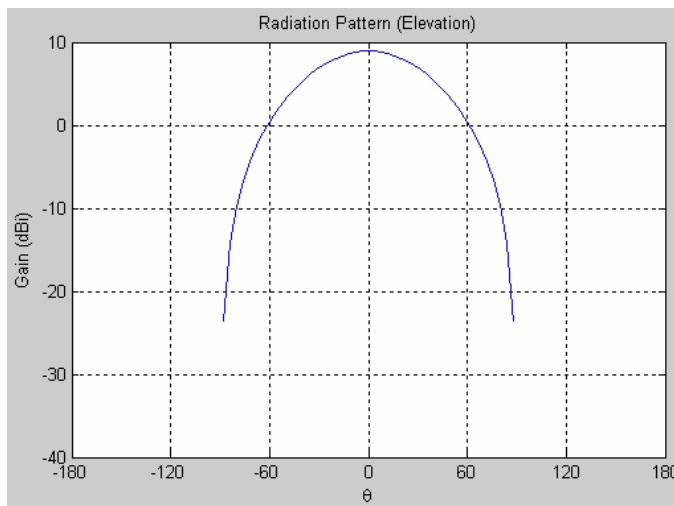


Figure 3.11 Double Quad "Bow Tie" Antenna; gain at 0° : 8.9 dBi, gain at 60° : 0.5 dBi

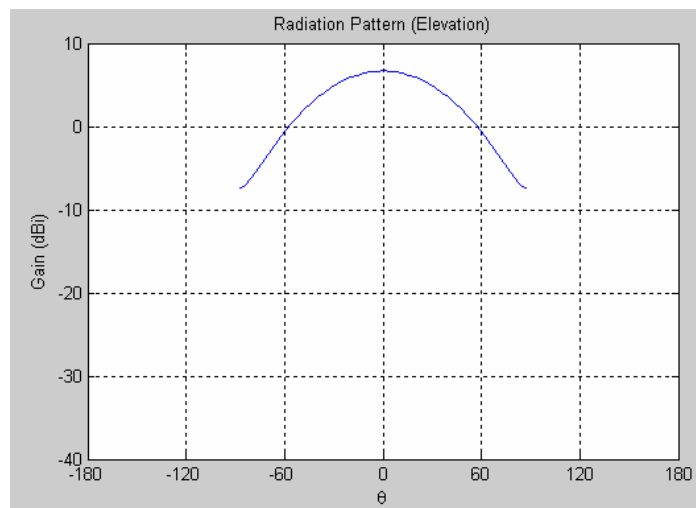


Figure 3.12 Rectangular Spiral Antenna; gain at 0° : 6.7 dBi, gain at 60° : -1.0 dBi

The Yagi antenna gave quite a good directional gain however as it is made of parallel elements it would likely get interference from nearby tracks on the PCB and hence in practice would not match the simulation. The mesh plate and rectangular loop antenna are quite flat near the normal ($\theta = 0^\circ$) so are not good for a directional design. The rectangular spiral does not drop off as much as the circular loop or the bowtie so was also discarded leaving the choice between the circular loop and the double quad bowtie. From these we chose the circular loop antenna for the first prototype badge design because the bow-tie is quite large in comparison (12.7cm end to end).

3.4 Final Antenna Design

As mentioned in section 3.1 after having selected an antenna design it was decided to change the SmartBadge design to have both an infrared and an RF communications link which meant the requirements of the antenna were changed. We now did not need a directional RF link as the badge-to-badge communication would be done with IR which naturally has a limited viewing angle. Based on this change we decided to employ the rectangular spiral design (Figure 3.6) for the antenna because it offered the largest gain over a wide angle. We also discovered that it performed much better without the ground plane behind it (the ground plane causes an image effect that is the equivalent of having a second antenna on the far side of the ground plane) so we removed the ground to get the radiation pattern shown in Figure 3.13 (the 3D radiation pattern is also shown in Figure 3.14).

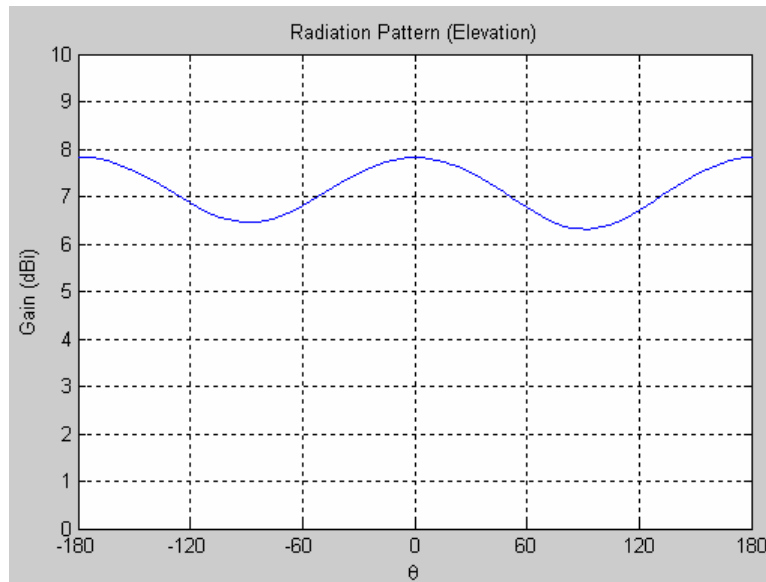


Figure 3.13 Rectangular Spiral Antenna without ground; gain at 0° : 7.8 dBi, gain at 60° : 6.8 dBi

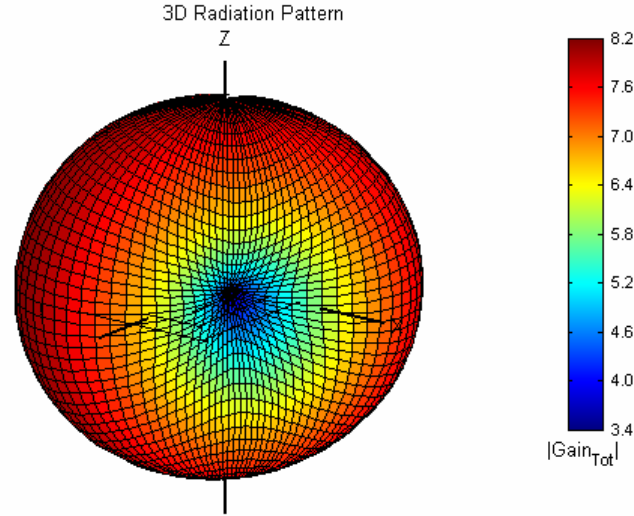


Figure 3.14 3D Radiation Pattern for Rectangular Spiral Antenna

3.4.1 Antenna Impedance

Another useful feature of SuperNEC [1] was that it is able to give the impedance of the chosen antenna design for a range of frequencies. At 868MHz SuperNEC claimed that the rectangular spiral had an impedance of $52 - j442 \Omega$. The CC1010 data sheet did not state what the output impedance of its RF transceiver was but the development kit had a matching circuit for a 50Ω load so we created a match between our antenna and this 50Ω matching circuit, as maximum power transfer occurs when the impedances are correctly matched.

3.4.1.1 Matching Network

The matching network we designed was a 3-element π -network using the combination approach [31]. A 3-element network has the advantage that we can choose a Q value (we arbitrarily chose a high value of 40) which is the quality factor used to measure stored/dissipated energy or bandwidth. The matching equations are as follows:

$$X_s = QR_s \quad (3)$$

$$X_p = X_s \left(1 + \frac{1}{Q^2}\right) \quad (4)$$

$$R_p = R_s (Q^2 + 1) \quad (5)$$

To create a 3-element network we use two 2-element networks (both matched to a virtual central resistance (R_s) as in Figure 3.15) and then join the series elements (X_{S1} and X_{S2}) together. To find the 2-element network at the antenna end we first find the

parallel equivalent of the antennas impedance and this gives R_{P1} (R_P in the above equations). Equation (5) then gives the value of R_S (different for each 2-element network) since we have specified Q . X_{S1} follows from equation (3). X_{P1} is found from equation (4) and is the parallel combination of the antennas reactance with that of the parallel reactance of the two element match. A similar process determines the 2-element network at the other end (except that the CC1010 is purely resistive so X_{P2} is just the reactance of the matching network), then R_S is removed and X_{S1} and X_{S2} combined to get the 3-element network as shown in Figure 3.16(a).

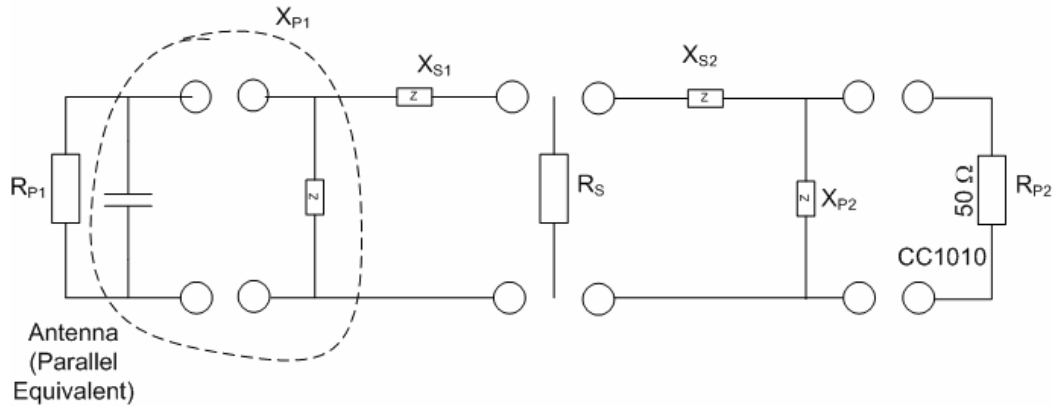


Figure 3.15 Two 2-element matching networks are used to create the 3-element π network

When the circuit is operating at 868MHz this results in the components and values shown in Figure 3.16(b) given that the reactance of a capacitor and inductor are given in equations (6) and (7) respectively where ω is the frequency.

$$X_C = \frac{1}{\omega C} \quad (6)$$

$$X_L = \omega L \quad (7)$$

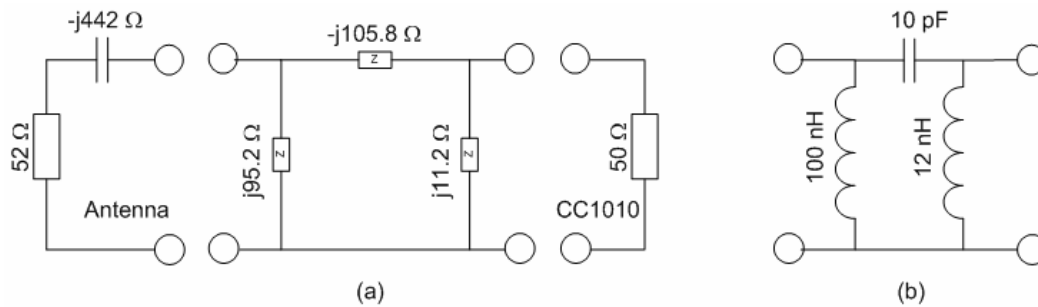


Figure 3.16 (a) Impedance Matching Network

(b) Component Values at 868 MHz

The bandwidth of this matching network is 21.7 MHz. This is found using equation (8), given that the Q value was chosen to be 40, and the frequency (ω) is 868MHz.

$$B = \frac{\omega}{Q} \quad (8)$$

3.4.1.2 Network Analyser

Once we had built the antenna we decided to attach it to a network analyser to measure the actual impedance and compare it to the generated value. In order to make the connection we had to solder on a short piece of coax cable as the network analyser needed a BNC connection to the antenna.

According to the network analyser the impedance was closer to $90 + j20 \Omega$ and hence we designed a new matching network for this impedance hoping to improve the transmission range of the radio. The range at the minimum output power of the CC1010 (-20dBm) was about 3m with the original matching network based on the calculated values, however when we used the measured impedance we got a range of only 0.5m. This means that the length of coax we had to add had a significant effect on the antennas impedance and hence we returned to the original matching network of Figure 3.16(b).

3.5 Chapter Summary

In this chapter we have designed the antenna for the SmartBadge's RF transceiver. Having identified the design goals a number of different possibilities were simulated with the chosen final design being that of Figure 3.6 with the ground plane removed. The matching network of Figure 3.16(b) was also developed to get the maximum power possible through to the antenna.

Chapter 4 SmartBadge Design

Having discussed the design of the antenna in the previous chapter this chapter provides an overview of all the other aspects of the hardware and design of the SmartBadge. It includes brief descriptions of the Chipcon CC1010 IC, which comprises the core of the SmartBadge, and also the peripherals attached to the CC1010. We then present the final design.

4.1 Chipcon CC1010

The HITLab already had some Chipcon CC1010 ICs and a development kit available so it made sense to use these as the core of the SmartBadge. The CC1010 contains an 8051 based microcontroller and an RF transceiver in the one package so is considered a System on Chip (SoC) solution. This section gives a brief overview of the important features, and more detail can be found in the datasheet [32].

4.1.1 8051 Microcontroller

The microcontroller used is based on the industry standard 8051 architecture. It has 32kB of on board flash memory as well as two SRAM banks of 2048 and 128 bytes. The peripherals include 3 Analogue to Digital Converters (ADCs), 4 Timers (including 2 Pulse Width Modulators (PWMs)), 2 UARTs (Universal Asynchronous Receive Transmit), an RTC (Real Time Clock), Watchdog Timer, SPI (Serial Peripheral Interface), Data Encryption Standard (DES) encryption/ decryption and 26 general purpose I/O pins. We have used most of these peripherals in the SmartBadge design. A block diagram of the entire CC1010 is shown in Figure 4.1.

4.1.2 RF Transceiver

The RF Transceiver is user programmable to operate at frequencies between 300 and 1000 MHz; however Chipcon recommends using the ISM (Industrial, Scientific, and Medical) bands at 315, 433, 868 or 915 MHz. We have hence chosen 868 MHz as it is an available band in the New Zealand radio spectrum and being in an ISM band means we do not require a license to operate the SmartBadge. The RF section is shown in the block diagram of Figure 4.2.

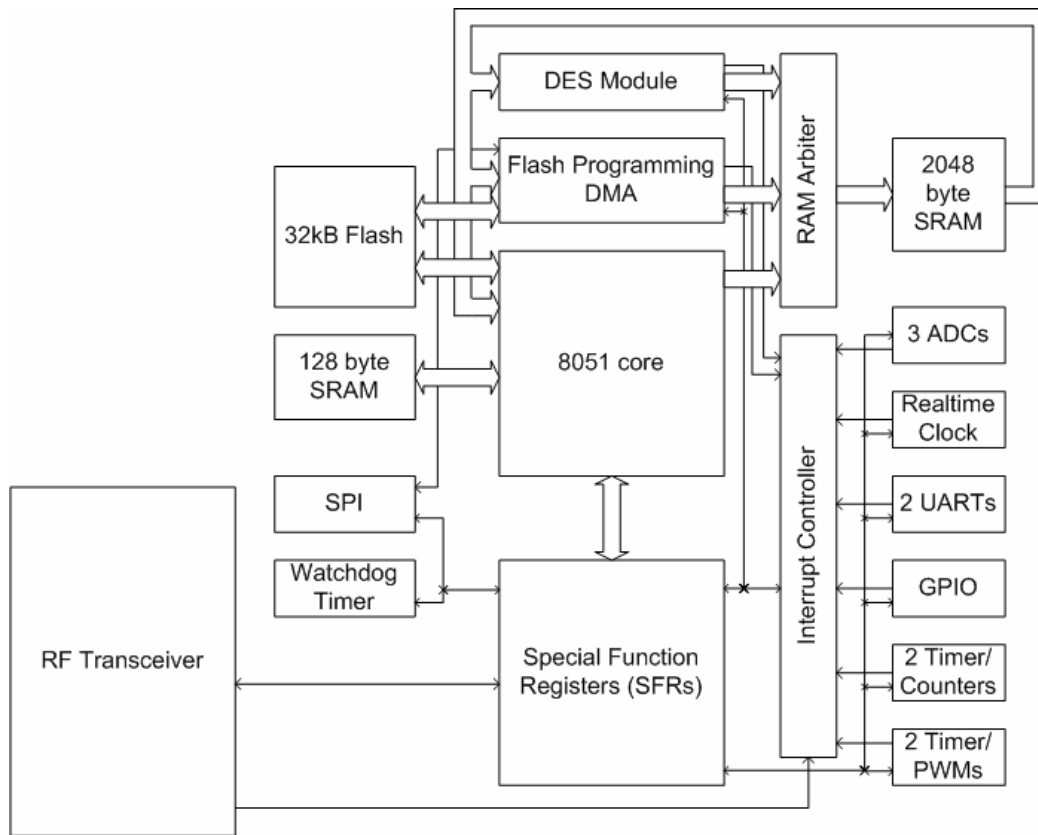


Figure 4.1 CC1010 Block Diagram [32]

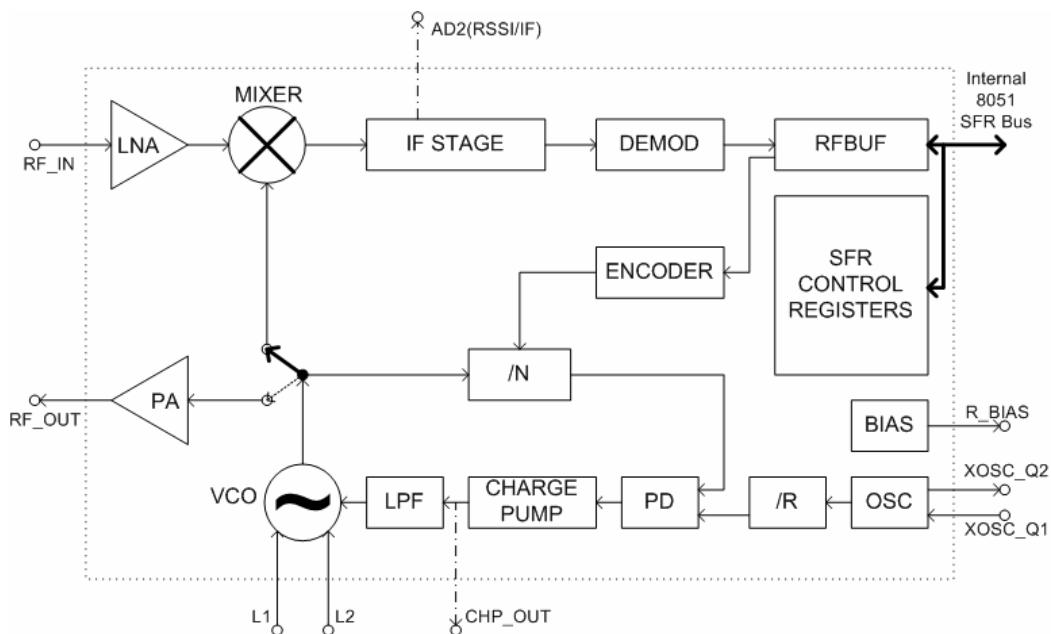


Figure 4.2 Block Diagram for RF Transceiver [32]

The RF modulation is BFSK (Binary Frequency Shift Keying) with a frequency separation programmable up to 65 kHz (we use the recommended 64 kHz). There are

a few different data formats that can be used but we have chosen to use synchronous Manchester coding at a symbol rate of 19200 baud (i.e. a data rate of 9600 bps); the maximum possible is 78600 baud. Chipcon strongly recommends either Manchester coding or NRZ (Non Return to Zero) as the other modes bypass the built in data decision circuitry. We have chosen Manchester coding as it has an equal number of highs and lows (and thus a zero DC value) which is a requirement of the preamble detection circuitry. Manchester coding is where a single bit is coded as a pair of pulses; a high '1' bit is a high frequency (i.e. 868 MHz + 32 kHz) followed by a low frequency (868 MHz – 32 kHz) and a low '0' bit is the reverse as shown in Figure 4.3. This means that each bit is two symbols hence the data rate is half the baud rate.

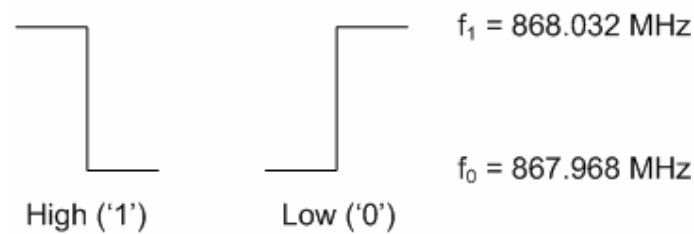


Figure 4.3 Manchester Coding

The output power ranges from -20dBm to 4dBm (at 868 MHz) and the receiver sensitivity is -106dBm. The transceiver uses an internal IF frequency of 130 kHz and can measure RSSI between -105dBm and -60dBm. For other details refer to the datasheet [32].

4.2 Peripherals

There are a number of peripherals attached to the CC1010, the most important of which is the infrared transceiver.

The IR transceiver is connected to a UART on the CC1010 via an interface IC which converts data from the UART (RS232) standard to the IrDA's SIR (serial infrared) standard. The differences between the two are that the SIR standard inverts the data and also changes the pulse width so that a high output is only 1.627μs long irrespective of the data rate, which conserves power.

There is a 16x2 character LCD screen included which is the main user interface on the SmartBadge. It connects to the CC1010 on the SPI interface by going through an 8-bit shift register. The LCD is primarily used to display the users name but can also display other information that an application may want. There are a couple of pushbuttons located beside the LCD display which can be used for example to scroll

through data should the application need to display more than can fit on the screen at once.

The other peripherals are four bi-colour LEDs that each have separate I/O lines for red and green and which could be used for an affinity score, and a piezoelectric buzzer running from the PWM module which makes a single tone noise should an application require it.

More details of these components are given in Appendix A including references to the relevant datasheets.

4.3 Final Design

Figure 4.4 is a block diagram of the final SmartBadge design. The antenna is connected to both RF_IN and RF_OUT via matching circuitry (see Figure 3.16(b)) as the CC1010 implements an RX/TX switch.

The infrared transceiver sends and receives data via the interface chip and also has its power supplied from the interface chip as it draws too much current to come from the CC1010 directly. The interface chip itself also has a reset pin and another to toggle between programming the baud rate and sending data.

The LCD gets its data from a shift register as there were not enough I/O lines on the CC1010 to be able to interface it directly. The pushbuttons are connected to the external interrupts as the CC1010 is in idle mode as often as possible so we could not automatically poll them.

The buzzer has to be run from the PWM as it needs a 4 kHz square wave to create a noise. The LEDs are basically a red and a green diode in one package and hence each needs two I/O lines to operate.

For full details of the design, including the schematic and PCB layout see Appendix A.

The final prototype is also shown in the photograph of Figure 4.5, and measures about 10cm by 10cm. It has the CC1010 and the programming connector on the reverse, with all the peripherals located on the front of the SmartBadge. The infrared transceiver achieves a range of 2m with the $\pm 24^\circ$ viewing angle, while the RF transceiver has a range of 3m (at minimum output power) over a greater than $\pm 60^\circ$ field of view. More details of prototype construction are in Appendix B.

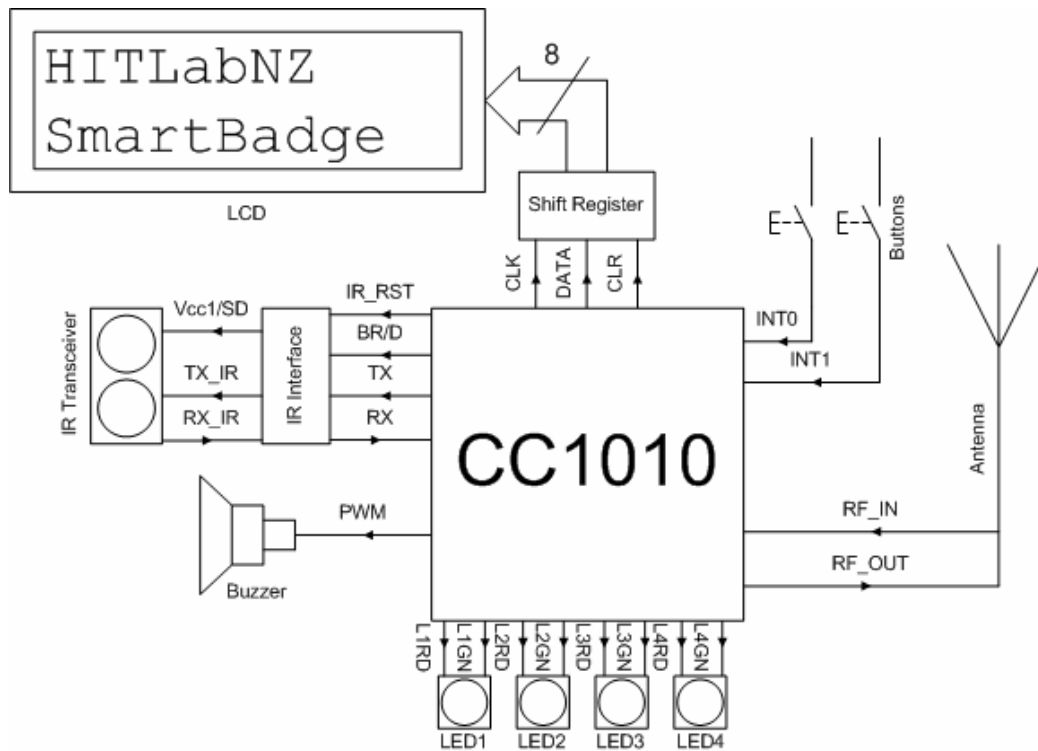


Figure 4.4 SmartBadge Block Diagram

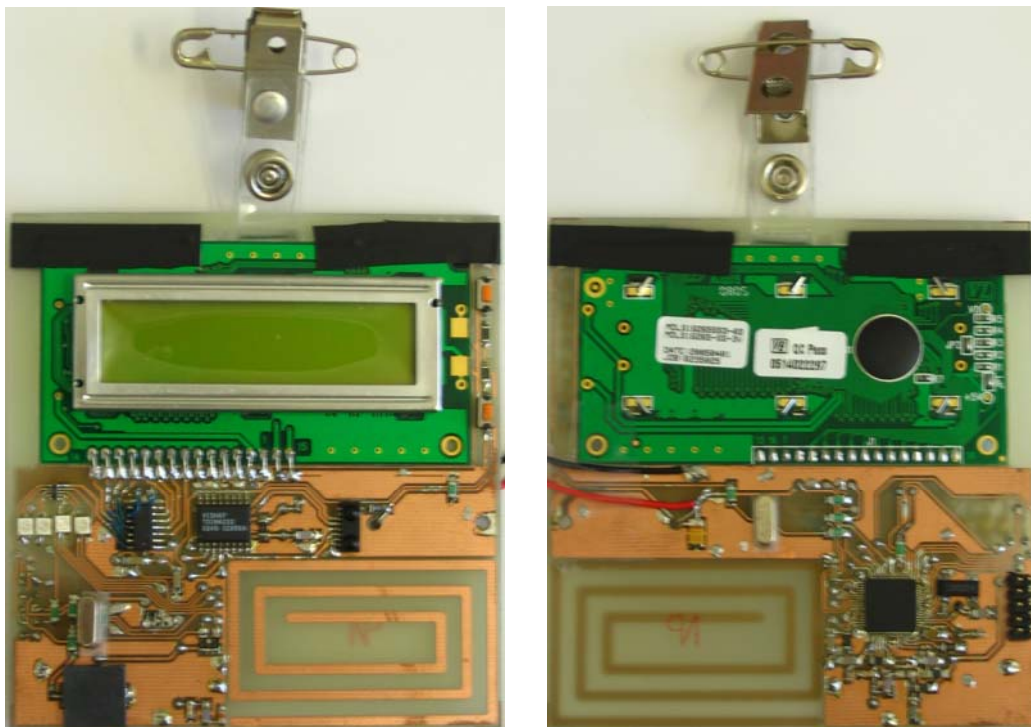


Figure 4.5 The final SmartBadge design, front and back

4.4 Chapter Summary

This chapter has described all the hardware components of the SmartBadge and how they interact. The SmartBadge is centred on the CC1010 microcontroller and RF transceiver IC. It also has a number of peripherals attached to the CC1010 including the infrared interface chip and transceiver, the LCD display, four LEDs, two buttons and a piezoelectric buzzer. These are connected to the CC1010 as shown in Figure 4.4, with the final SmartBadge PCB photographed in Figure 4.5.

Part II:

Software

In the software section we describe how the SmartBadges communicate. This discussion begins with Chapter 5 which details the badge-to-badge protocol, where communication is via the infrared transceiver. The badge-to-badge protocol is used to collect the initial data from other SmartBadges a user comes in contact with.

Next, in Chapter 6, we consider the badge-to-base RF section which looks at two different ways of transferring the collected data back to a central server for processing and storage.

The first approach considered was the use of routing algorithms to direct the data through the SmartBadges as a multi hop ad hoc mobile network to a single access point. Simulations for this approach are presented along with a discussion of its strengths and weaknesses.

The second method that was considered was the use of multiple access points distributed around the venue such that any SmartBadge is within reach of at least one access point. This required the design of a new Medium Access Control Protocol (U-MAC), which is described in the second half of the badge-to-base chapter along with simulations of this new protocol. This approach had the advantage of not requiring any multiple hop communication.

Chapter 5 Badge to Badge Protocol

This chapter describes the communications between two SmartBadges which happens solely over the infrared link (RF is used to communicate with the base stations, see Chapter 6). The infrared devices we have used conform to the Infrared Data Associations (IrDA) serial infrared (SIR) specifications for the physical layer. This specification also dictates what format the data is sent in and is described in the first section below. The second section explains how this data format had an effect on the infrared hardware.

The major difference between the radio and infrared link is that infrared is a directional link as the infrared diode only transmits over a limited angle from its normal (the angle of half intensity for the TDFU4100 is $\pm 24^\circ$ [34]) meaning that interference from other SmartBadge pairs is limited. Also the transmission range of infrared is about 1 metre at maximum speed (around 2m at the speed the SmartBadges use) so in fact potential interference is negligible and it can be assumed that the medium will be idle unless a SmartBadge is transmitting or receiving.

Since the infrared system operates a directional point-to-point link only, there is no routing to be concerned with. This means that the only software developed for the badge-to-badge protocol is specifying the synchronisation of data and also what data is transferred. This is described in the third section.

5.1 Data Format

The data is sent from the CC1010 via its UART interface at a rate of 9600bps. The UART is operating in full duplex mode and sends a start bit (0) follow by 8 data bits (LSB first), a parity bit (even parity) and finally a stop bit (1). Full duplex operation cannot be utilised, however, as there is only a single output channel for the infrared transceiver; namely the air between the badges, as opposed to two separate wires for a conventional UART connection. The UART packet is shown in Figure 5.1 for the transmission of an ASCII 'U' (the byte 0x55). Note that the medium is in a logic '1' state when idle.

The IrDA standard will not accept data in this format though and hence the need for the TOIM4232 codec. The IrDA standard sends data in $1.627\mu\text{s}$ pulses which are inverted from the original data. This means that the data sent over the infrared link

actually looks like the packet shown in Figure 5.2 which greatly reduces the energy required to transmit the data.

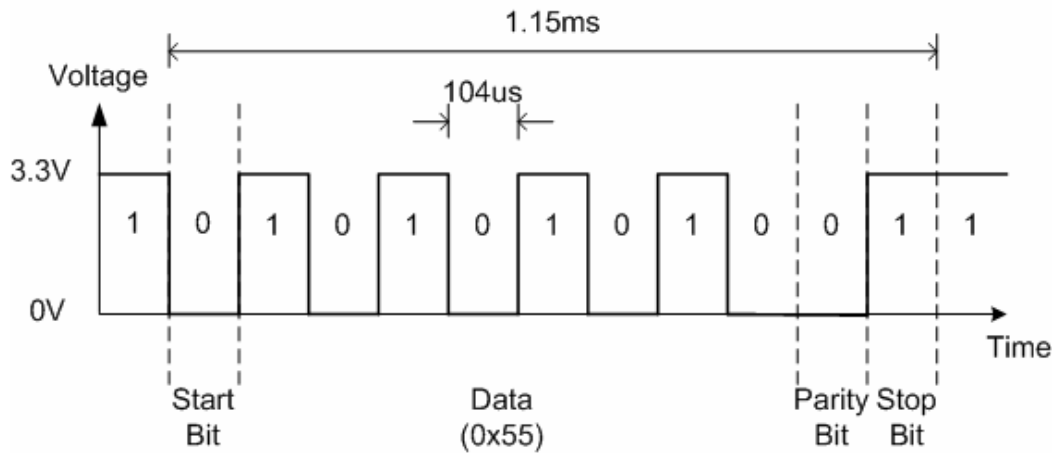


Figure 5.1 UART data format of the letter 'U'

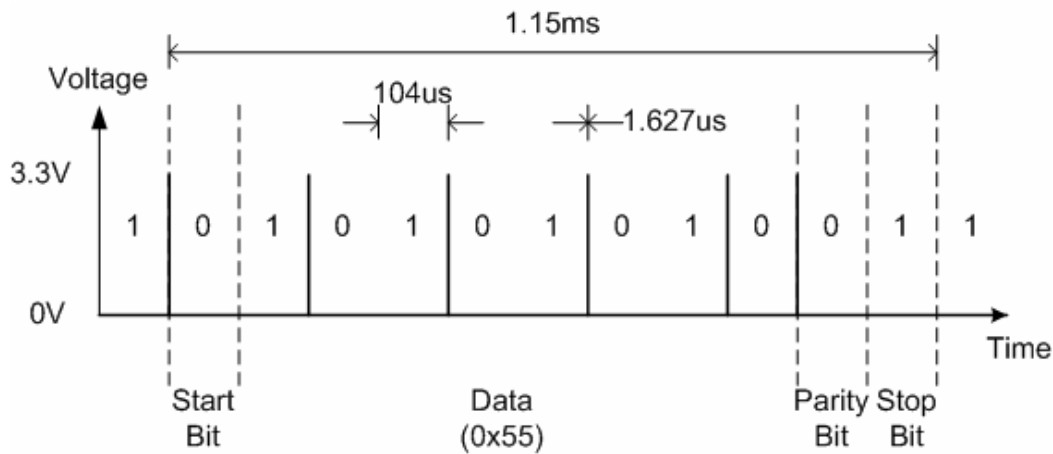


Figure 5.2 Infrared data format of the letter 'U'

5.2 Infrared Hardware Considerations

The infrared hardware did not work at the first attempt. We realised that this was because the other IC mentioned in the TDFU4100 Infrared Transceivers datasheet was not a RS232 level converter from the same company, as we had originally thought, but actually a necessary interface chip since the infrared transceiver uses the IrDA's SIR data format and not the RS232 serial port standard that the CC1010 uses to send data.

Once we added this TOIM4232 codec³ the infrared transceiver did work successfully over the angles specified in the datasheet (+/- 24°) at a range of about 2m. This translates to one SmartBadge being able to communicate with another provided the

³ The TOIM4232 datasheet [33] refers to it as an endec (encoder / decoder)

height difference between them is less than 80cm (at 2m apart) which would be likely when the SmartBadges are worn by people.

5.3 Badge-to-Badge Software

When a SmartBadge is turned on it begins searching for a partner to communicate with. It does this by transmitting its ID over the infrared interface continuously 5 times (5.73ms total), then listening for 5 data periods and repeating this continuously as shown in Figure 5.3.

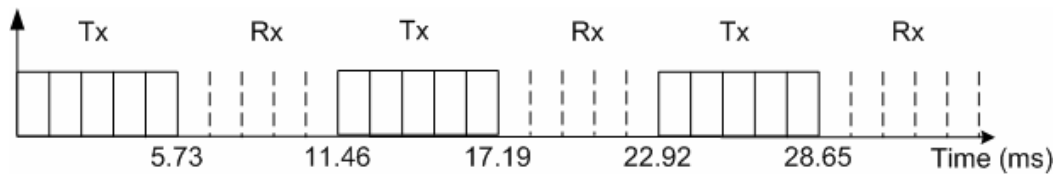


Figure 5.3 SmartBadge partner search timing

Once a SmartBadge receives something (i.e. a second SmartBadge is in range) it begins synchronisation by stopping its transmissions and waiting until the same packet is received three times in a row, or a timeout occurs in case of error (see the state diagram of Figure 5.4). This is because a valid packet can still be found as long as it has a start bit (0), 8 data bits, a valid parity bit and a stop bit (1) even if it is not the intended packet and this often happened in our testing. Getting the same packet three times meant that, with high probability, it was actually the correct one. This is similar to repetition coding where the likelihood of error is reduced by sending the data multiple times and making a majority decision.

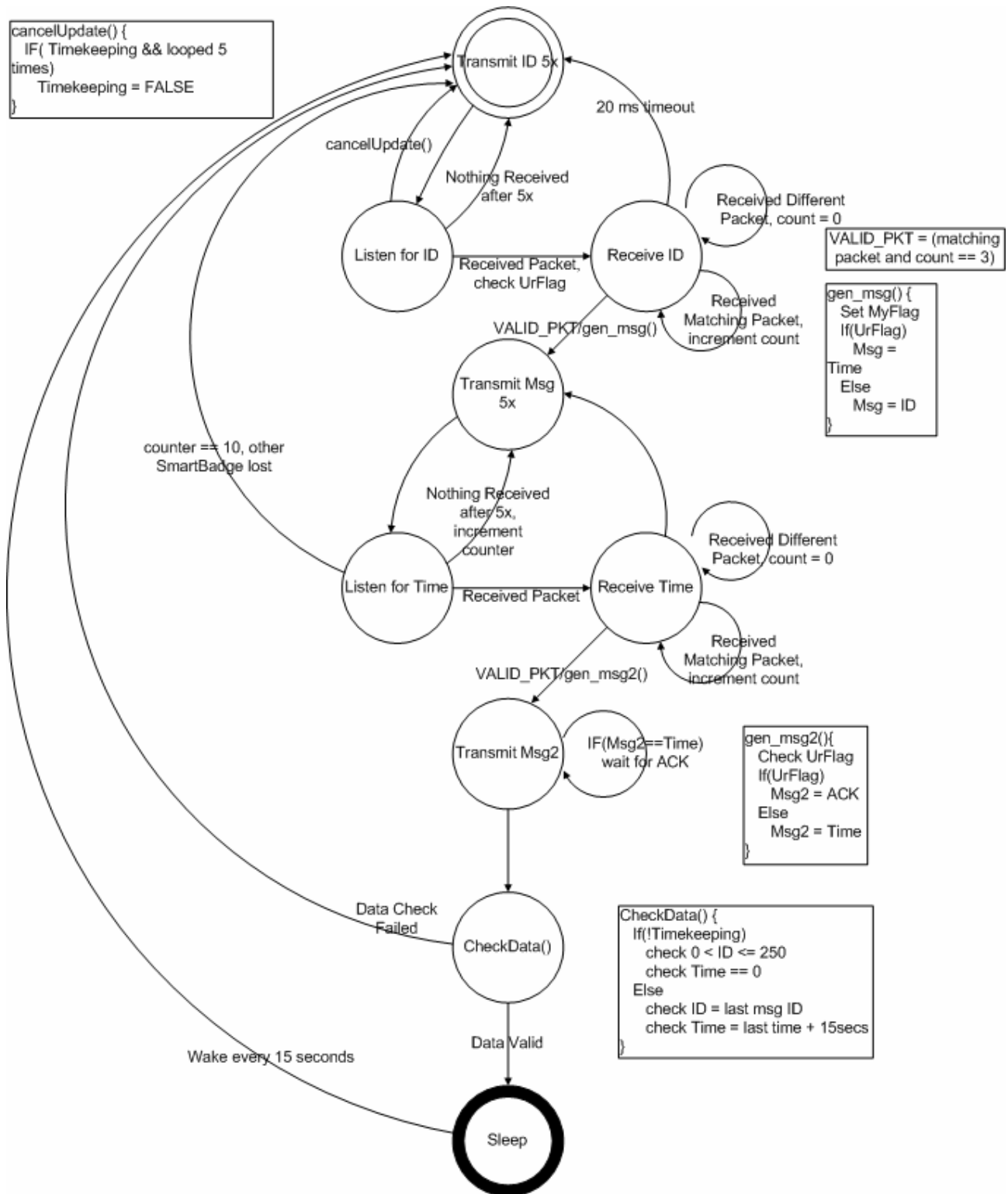


Figure 5.4 Badge-to-Badge Protocol State Diagram

Having received this packet (the sender's ID) the SmartBadge replies with its own ID continuously (or until timeout) so that the second SmartBadge can itself get the ID three times. Once the second SmartBadge gets this ID the same exchange is done for the time counter (which should be zero at this stage).

Once both SmartBadges have each others ID and time variable, error checks are made on these. In the business card exchange application developed here we have chosen to use an 8-bit ID number so this means the ID must be between 1 and 250 (251-255 are reserved should an application require the base station to have IDs, and zero could cause confusion with the time variable) and the time must be zero. The ID could easily be larger than 8-bits if necessary.

After the SmartBadge has checked for valid data it goes into the timekeeping phase. When in timekeeping the IR transceiver is turned off most of the time. It will wake up every 15 seconds in order to check that it is still within range of another SmartBadge. When it wakes it will go through the transmit/receive cycle of Figure 5.3 five times to try and hear an updated message from the current SmartBadge. If it receives this message it will check for errors and then go back to sleep; the ID should be the same as in the previous exchange, and the time should be one interval greater than the stored value. If it does not receive an update message it will assume that the current conversation has ended and hence will go back to the initial search mode.

5.4 Chapter Summary

This chapter has described the operation of the badge-to-badge protocol. It operates via the infrared transceiver and so the reason for needing the codec is explained (the reason being a difference in expected data formats). The protocol itself is also described and is summarised in the state diagram of Figure 5.4.

Chapter 6 Badge to Base Protocol

This chapter describes the protocol the SmartBadges use when communicating with the base stations to upload the data they have collected over the infrared link. It starts by describing the routing protocol approach that we had intended to use but due to the shortcomings discussed in section 6.1.3 we have designed a new medium access control protocol which is described in section 6.2. This is then followed by the simulations of this new MAC protocol.

6.1 Routing Protocols

The original plan was to design a network based around a single central base station to which all nodes would send data. This meant that nodes not directly in range of the base station would have to route their data through other nodes and hence we investigated wireless routing protocols using the NS-2 [35] network simulator from the Information Sciences Institute at the University of Southern California. We started work based on an article by Royer and Toh [36] which discusses different routing protocols for wireless ad-hoc networks. It classifies the protocols into two broad types: table driven and source initiated.

A table driven protocol is one that keeps records of the route from each node to any other node in the network. Changes in network topology cause these records to be updated so as to keep a consistent network view. The examples of table driven protocols presented were Destination Sequenced Distance Vector (DSDV) Routing [37], Clusterhead Gateway Switch Routing (CGSR) [38] and the Wireless Routing Protocol (WRP) [39].

The other protocol type is source initiated or on-demand routing. This type only creates routes when the source node needs them. When a node requires a route it starts a route discovery process which will either find a route or exhaust all possibilities. This route is then maintained via a maintenance procedure until there is no longer a path to the destination (due to node mobility) or the source no longer needs the route. Examples of source initiated protocols are Ad-hoc On-demand Distance Vector (AODV) Routing [40], Dynamic Source Routing (DSR) [41], Temporally Ordered Routing Algorithm (TORA) [42], Associativity-Based Routing (ABR) [43] and Signal

Stability Routing (SSR) [44]. A couple of these protocols are briefly evaluated in section 6.1.1.

6.1.1 Simulations

From reading the relevant papers about the above protocols we discovered NS-2 [35], a free open source network simulator that was used to simulate the protocols in a number of the papers and so we decided to use it to create our own protocol simulations.

NS-2 already implemented four of the above protocols, DSDV, AODV, DSR and TORA so we decided to try these first. Unfortunately the DSR and TORA implementations were not complete so we started with DSDV and AODV which were examples of a table driven and a source initiated protocol and thus were a good starting point. The details of DSDV and AODV are described in Appendices C.1 and C.2 respectively.

The simulations therefore tested both DSDV and AODV over four different room configurations with radio ranges of either 3 or 5 metres and first with just one node transmitting, then all of them. The four different rooms are summarised in Table 6.1 and the 10 x 10 metre room is shown in Figure 6.1. Note that the nodes are randomly located and move around with time. Also node 1, in the centre of Figure 6.1, is the fixed base station.

Scenario	Room Size (m)	Number of Nodes
1	5 x 5	10
2	10 x 10	20
3	20 x 20	150
4	20 x 20	200

Table 6.1 Room sizes and node counts for routing scenarios

The other parameters used were the two ray ground propagation model, IEEE 802.11 [16] MAC layer, omni-directional antenna, and a maximum queue length of 50 packets.

The simulation was run for 600 seconds but 1 second of simulation time represents 1 minute of real time (i.e. the data rate and power usage were adjusted accordingly) so this is equivalent to 10 hours. The traffic model is a UDP/CBR (User Datagram Protocol / Constant Bit Rate) connection that sends a 10 byte packet once every 5 seconds (minutes).

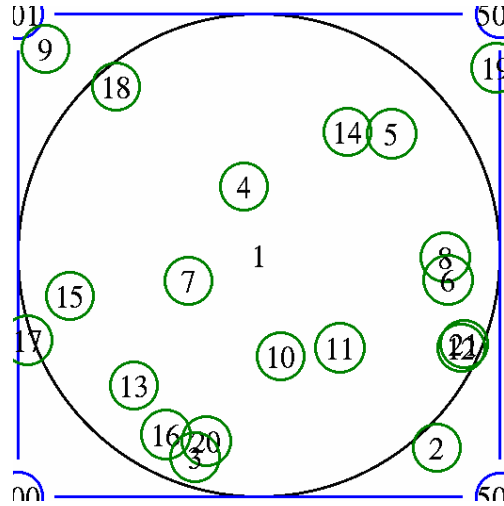


Figure 6.1 10 x 10 metre room, 20 nodes (green), range 5m (black ring is range of base station)

The energy values are based on using a single CR2032 coin cell battery which has 2376J of energy (220mAh at 3V). The power required transmitting (3m and 5m), receiving and while idle was 246.3 mW, 247.5 mW, 256.2 mW and 243 mW respectively. These values are comprised as shown in Table 6.2 which shows that most of the SmartBadges power requirements are from the peripheral devices and not the CC1010 itself. The full simulation script is included in Appendix B.3.

Item	Power Consumption (mW)
4 x LEDs	120
LCD (no backlight)	18
IR transceiver and endec	24
Shift register for LCD	45
TOTAL PERIPHERAL POWER	207
CC1010 idle, RF in RX	36
CC1010 active, RF in RX	49.2
CC1010 active, RF in TX (3m)	39.3
CC1010 active, RF in TX (5m)	40.5

Table 6.2 Power consumption breakdown for routing simulations

6.1.2 Results

The results we were most concerned with were the network lifetime and the message throughput. These results are presented below.

6.1.2.1 Network Lifetime

The most important factor to consider is how long nodes will survive under each routing protocol, i.e. how energy efficient they are. These results are presented in Figure 6.2 and Figure 6.3 for one and all nodes transmitting respectively.

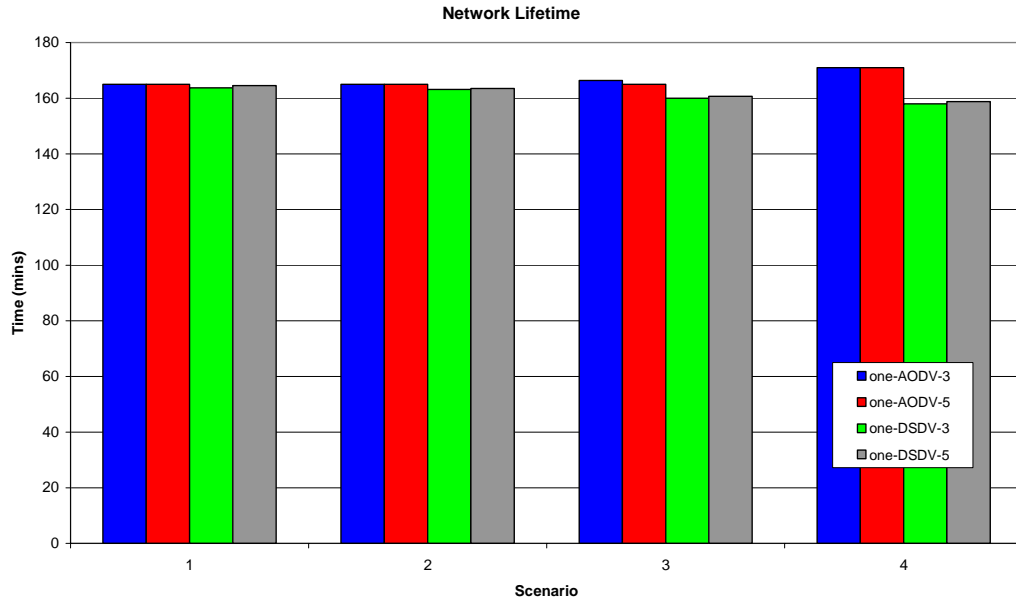


Figure 6.2 Network Lifetime with a single node transmitting

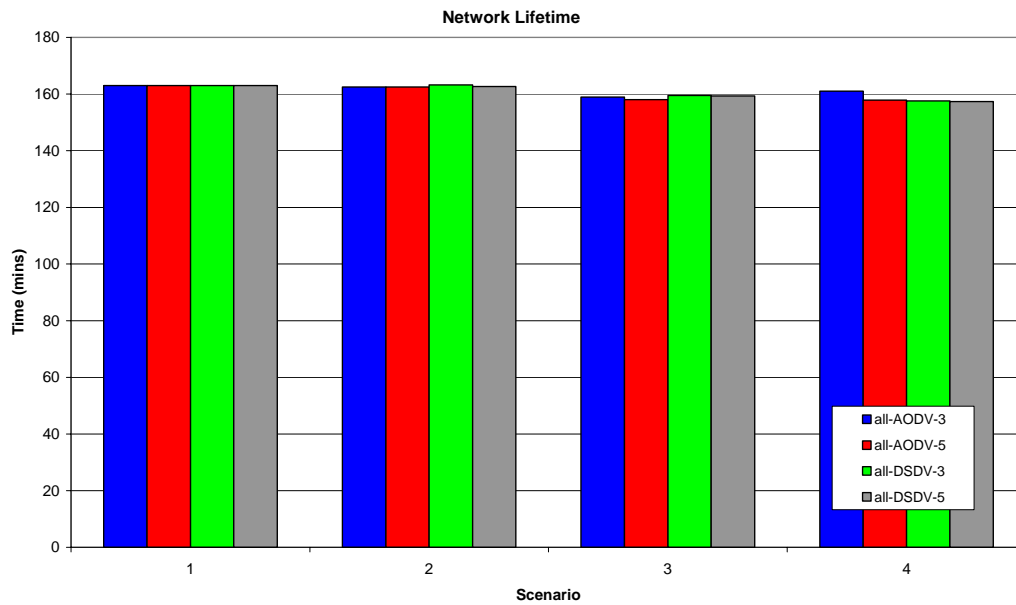


Figure 6.3 Network Lifetime with all nodes transmitting

The values presented are the average time at which all the nodes in the network die (remaining energy becomes zero). The legends mean how many nodes transmitting (one or all), which protocol was used (DSDV or AODV) and what radio range (3 or 5 metres). The scenarios across the bottom are those from Table 6.1.

There is little difference among the results (average lifetimes are approximately 160 minutes) as the main sink of energy on the badges is in fact the LED's. Therefore, we ran the simulations again without the LEDs (and hence we no longer needed the shift

register either as this freed enough I/O pins on the CC1010) and these adjusted values of energy consumption when receiving transmitting and idle increased the average lifetime significantly.

With the LEDs removed (Figure 6.4 and Figure 6.5 for one and all nodes respectively) the new power consumption values are:

- Transmitting (3m) 81.3 mW
- Transmitting (5m) 82.5 mW
- Receiving 91.2 mW
- Idle 78.0 mW

The nodes lifetimes consequently jumped to about 510 minutes (8.5 hours) for the smaller scenarios but achieved only around 440 minutes (7.25 hours) in the larger scenarios which would be expected as SmartBadges are then more likely to have to spend extra energy forwarding packets from other badges. The only major difference between DSDV and AODV occurs in the large scenarios with a single transmitter, the difference being reasonably significant at about an hour.

Unfortunately NS-2 is not a complete simulation product and hence we were unable to get data for the all-DSDV-3 simulation for scenario 4 in Figure 6.5 (and Figure 6.7).

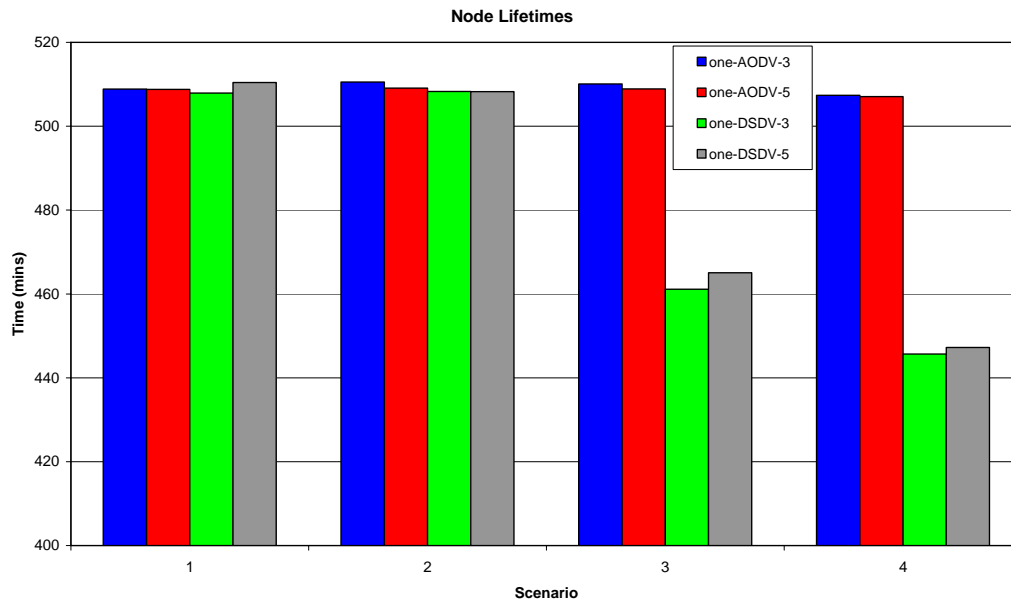


Figure 6.4 Node Lifetime with LEDs removed and a single transmitter

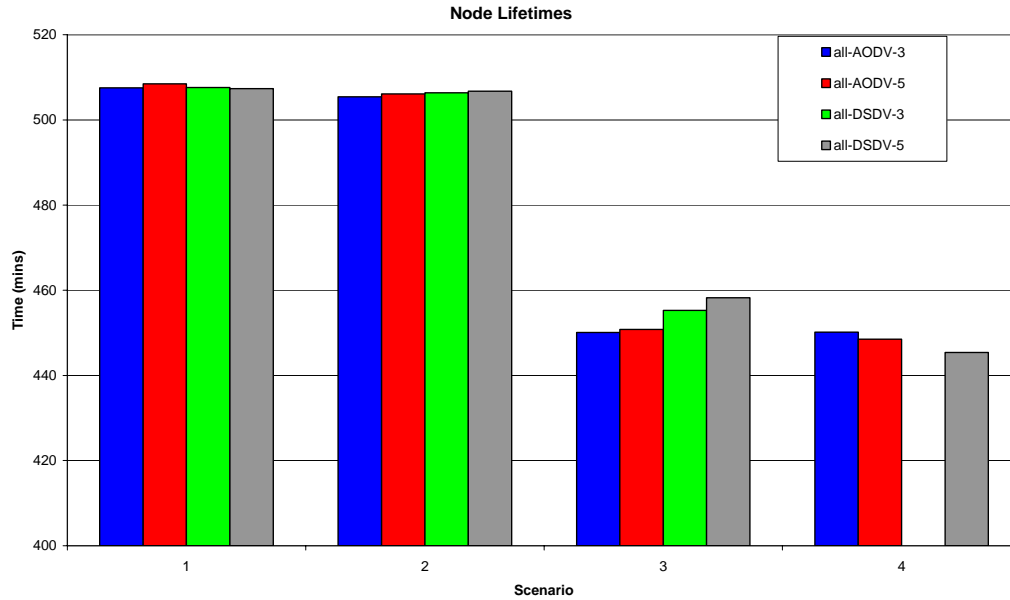


Figure 6.5 Node Lifetime with LEDs removed and all nodes transmitting

6.1.2.2 Message Throughput

The other results we collected were the number of packets sent and received by each node. These were collected for the case when the LEDs were removed as that meant the network was active for longer so we could gather more data. We used these values to determine what percentage of packets sent from the nodes were successfully delivered to the base station and present these in Figure 6.6 and Figure 6.7 below.

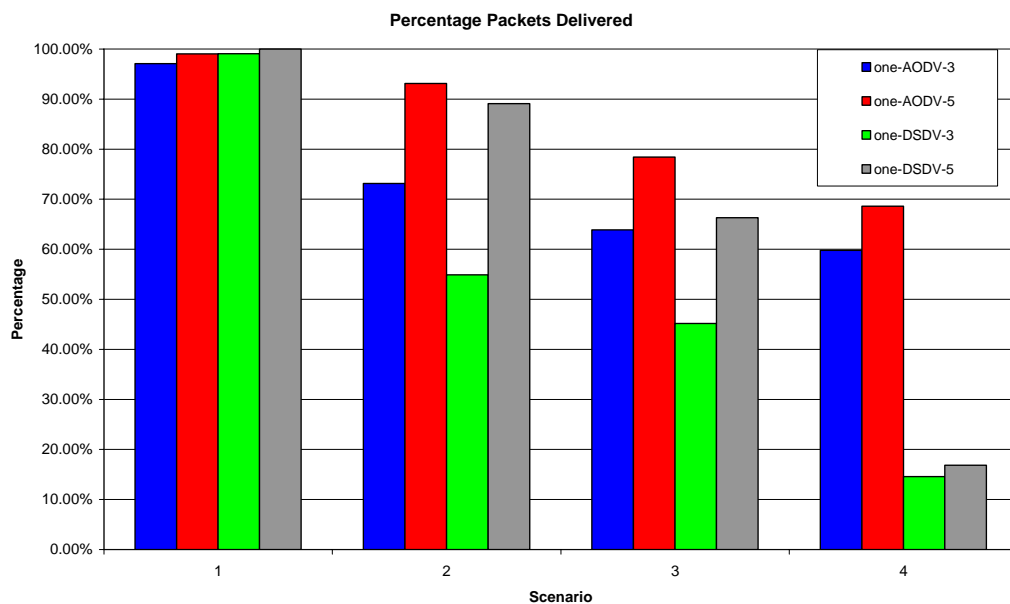


Figure 6.6 Packet Delivery Rate with a single transmitting node

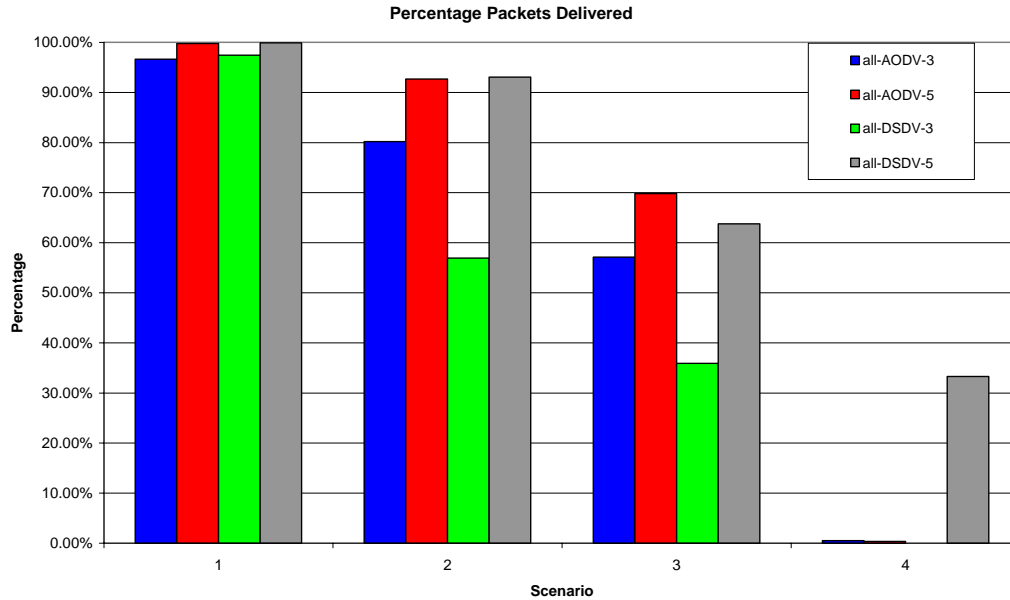


Figure 6.7 Packet Delivery Rate when every node is transmitting

From these graphs it can be seen that fewer packets are delivered for the 3m range than the 5m range which is to be expected (the reason for investigating the different ranges was to see if 3m was acceptable since it uses less power). For the first scenario almost all packets reach the base station but the delivery rate drops significantly as the room size and number of nodes increases. AODV outperforms DSDV in all scenarios except for the last one; the 20 x 20 metre room with all 200 nodes transmitting. This suggests that AODV may fall apart under high levels of traffic congestion but if high traffic congestion can be avoided AODV would be the best choice of protocol.

6.1.3 Analysis

The use of routing protocols means that only one base station is required as nodes can relay messages through other nodes to reach it. This means less cost is required to deploy the network hardware. However it can be seen in the graphs of throughput that when there are a large number of nodes the base station gets congested even though there is very little data being sent. It also means that nodes closer to the base station are having to do more work as they are not only sending their own messages but are also forwarding messages from more distant nodes and hence these nodes will die out quicker. It is for these reasons that it was decided to have multiple base stations

connected by means of a wired backbone⁴ network to a central database. This means an increased hardware cost but the base stations will not get over-congested. It also means that the base stations can be positioned so that the majority of the space available is covered and hence the nodes will be able to communicate with at least one base station directly so a routing protocol is no longer required. The focus of the badge-to-base protocol has therefore been changed from a routing problem to that of multiple access. This means we must now design a medium access control protocol to decide which node can talk to a base station at any given moment in time.

6.2 Medium Access Control Protocols

As a result of the weaknesses of the routing approach it was suggested that a preferred method was to have multiple base stations such that the SmartBadges see a single hop network. This meant that the design focus shifted to medium access control. This section describes a novel MAC protocol (Uplink MAC, or U-MAC) which is based on some of the features of the MAC protocols described in section 2.3. It includes a description of the network and traffic models U-MAC is designed for use with, the U-MAC protocol itself, and a comparison of U-MAC to other MAC protocols.

6.2.1 Network and Traffic Models

The first step in designing a protocol is to define the network and traffic models with which it is to operate. The SmartBadge network will consist of a large number of badges all within a single hop of a base station. The range of the SmartBadges is likely to be less than 10m (to conserve power) which covers an area of 314m^2 . If each user occupies about 1m^2 then there will be a maximum of around 300 SmartBadges per base station.

U-MAC is designed for scenarios where a collection of nodes are uploading a fixed amount of data at fixed time intervals to a central server. The traffic will be the data collected from the badge-to-badge protocol which, since it needs to be uploaded to a central database, will be primarily unidirectional towards the base stations (which are then wired to the central database). The badge-to-badge infrared protocol collects an 8-bit SmartBadge ID and an 8-bit time counter every 15 seconds. The badge-to-base protocol will send this data to the base stations every 2 minutes which means that the RF data payload will be at most 8 pairs of 16-bits which is a total of 128 bits (or 16

⁴ This could in fact be a wireless backbone (e.g. IEEE 802.11) as long as the base stations remain stationary and there is no radio interference with the SmartBadges themselves.

bytes). We add to this a source ID (8 bits) and a sequence number (6 bits) to get a total packet size of 142 bits, or $17\frac{3}{4}$ bytes, as shown in Figure 6.8.

badge_ID(8 bits)	seq_no (6 bits)
IR_data_0.ID(8 bits)	IR_data_0.Time(8 bits)
IR_data_1.ID(8 bits)	IR_data_1.Time(8 bits)
IR_data_2.ID(8 bits)	IR_data_2.Time(8 bits)
IR_data_3.ID(8 bits)	IR_data_3.Time(8 bits)
IR_data_4.ID(8 bits)	IR_data_4.Time(8 bits)
IR_data_5.ID(8 bits)	IR_data_5.Time(8 bits)
IR_data_6.ID(8 bits)	IR_data_6.Time(8 bits)
IR_data_7.ID(8 bits)	IR_data_7.Time(8 bits)

Figure 6.8 U-MAC Data Packet Format

The symbol rate used is 19.2 kBaud and so since we use Manchester encoding the bit rate is 9600 bps. This means the time taken to send the data packet is $142 / 9600 \approx 14.79$ ms. Since there is 2 minutes between updates this means the radio transceiver in each SmartBadge should be idle for all but 14.79ms out of every 2 minutes (i.e. approximately 99.99% of the time) and so using a sleep/wakeup cycle similar to that of S-MAC will save significant energy.

A second type of packet used is the control packet (Figure 6.9) which also has the SmartBadges 8-bit ID number (only one ID is needed since the other is always a base station), and 6-bit sequence number. It also has a 2-bit type field which is defined as follows:

- ‘01’ RTS Packet
- ‘10’ CTS Packet
- ‘11’ ACK Packet
- ‘00’ Cancel (used by server when 2 base stations both hear the same packet)

badge_ID(8 bits)	
seq_no (6 bits)	type(2 bits)

Figure 6.9 U-MAC Control Packet Format

6.2.2 Uplink MAC (U-MAC) Protocol

All direct RF communication in the network is between a SmartBadge and a base station (if there was ever a need to send messages out to a SmartBadge this could be done as part of the ACK packet rather than a broadcast). The base stations are not mobile and hence can be mains powered and do not need to go to sleep. This means that when a node (SmartBadge) wakes up it knows the receiver will be awake and

hence it doesn't need to learn the sleep schedules of its neighbours other than to find a free time slot to send data.

The U-MAC protocol is shown in state diagram form in Figure 6.10. This diagram comes from OPNET (see section 6.3) where the **INIT** state is used to set the range and register statistics, so things really begin in the **SLEEP** state.

When in the **SLEEP** state the radio should be turned off (to achieve this in simulations all received packets are immediately destroyed) until a wakeup interrupt is received. Upon receiving this interrupt the SmartBadge generates an RTS packet (which is transmitted as soon as the carrier sense says the medium is free) and moves to the **RX_CTS** state to await a CTS reply.

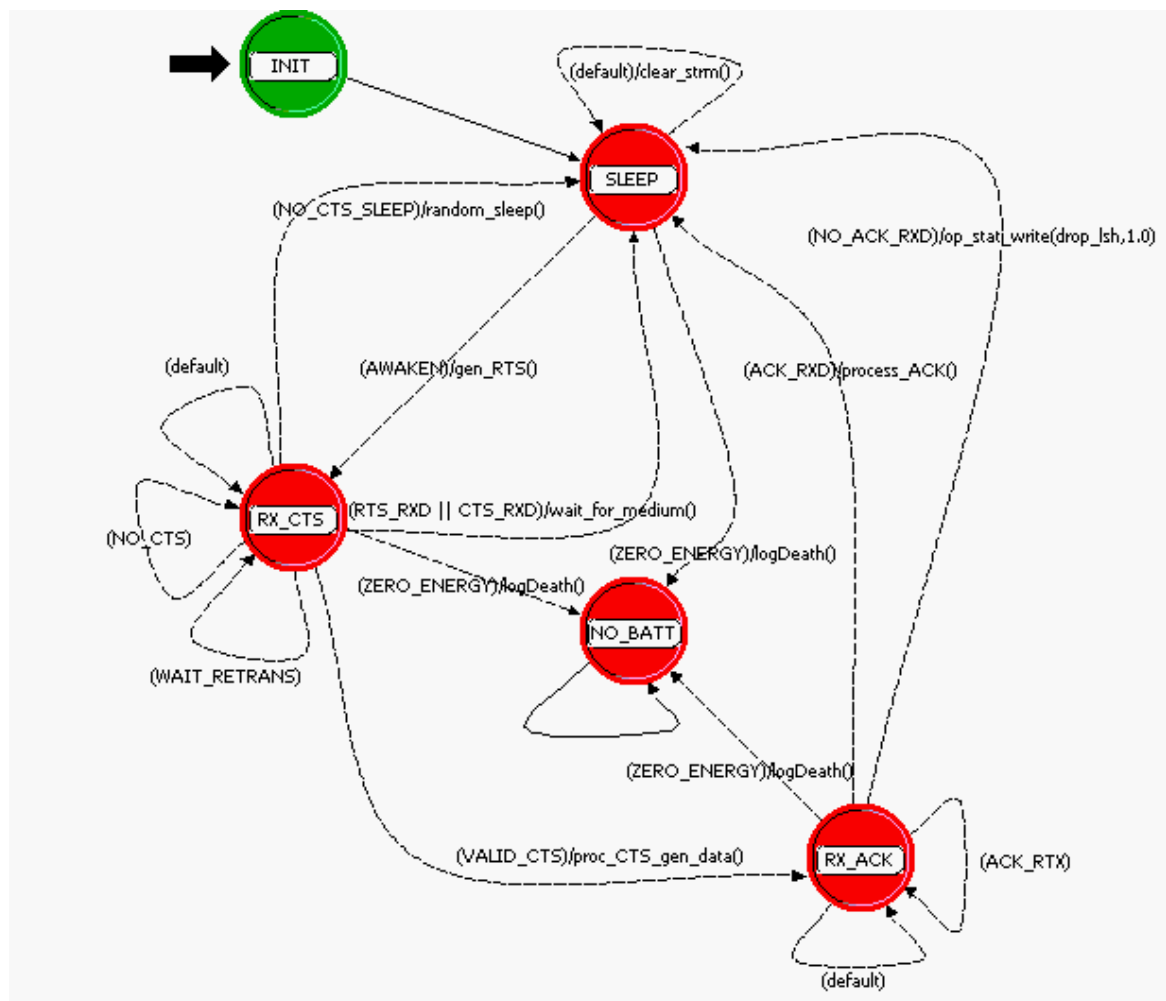


Figure 6.10 U-MAC State Diagram from OPNET [45]

Upon entering this state a timer is started so the possible interrupts here are a new incoming packet or a timeout. If a new packet is received it is checked to see if it is the CTS reply we are expecting and if it is we then generate a data packet and move to

the **RX_ACK** state. If however the packet is an RTS or CTS related to another SmartBadge then we must wait for the medium by sleeping for a short interval (the time taken for the other SmartBadge to complete its data transfer plus a short random time) and so go back to the **SLEEP** state.

If we have a timeout rather than a new packet then we follow the NO_CTS path and try again to receive a CTS, up to a maximum of 3 timeouts. If after 3 timeouts there is still no CTS then we follow the NO_CTS_SLEEP path back to the **SLEEP** state and sleep for a random interval (less than two minutes) as the most likely cause is collisions at the base station so we must find a new slot to transmit in.

If each message transfer takes 19.79ms (16bits (RTS) + 16bits (CTS) + 142bits (data) + 16bits (ACK) = 190 bits @ 9600 bps \approx 19.79ms) then it should be possible to transmit messages from approximately 6060 SmartBadges (2 minutes divided by 19.79ms since the SmartBadges will have new messages in 2 minutes time). As there should be no more than 300 SmartBadges per base station this means there is unlikely to be a problem finding an idle time slot before more data is queuing to be sent.

Following a successful CTS reply we prepare the data and enter the **RX_ACK** state. As we have now won the medium through RTS/CTS exchange there is no need to retry on an ACK timeout so in both cases we return to the **SLEEP** state. If an ACK was received this is recorded and the data can be cleared from the SmartBadges' memory as it has been successfully uploaded. If however there was no ACK this is recorded as a dropped packet and the data is kept for the next update (which will still be the same size as a user is highly unlikely to be meeting new people every 15 seconds so the data can be combined and only the latest time for each contact transmitted).

The **NO_BATT** state and the WAIT_RETRANS and ACK_RTX loops of the **RX_CTS** and **RX_ACK** states respectively are all described in section 6.3.1.1.

6.2.3 Comparison to Other Protocols

One point developed by the S-MAC [15] protocol is its message passing feature for handling messages greater than a packet length. As all the messages uploaded with U-MAC will be small (16 bytes maximum payload) there is no need to handle fragmentation of large messages. However should an application require data to be sent out from the base stations (this data could well be larger than an uploaded packet) it should reserve the medium for the whole message by sending the length of the data

to be sent in the CTS packet and the remaining length in the ACK packets the SmartBadge sends back. This way any SmartBadge that hears these will know to sleep until the message should be finished. Since there is so much free time between SmartBadge updates a longer message should not affect other badges upload schedules dramatically.

In the other MAC protocols (specifically T-MAC [17], CSMAC [22], DSMAC [23] and PMAC [24]) identified in section 2.3, putting a node to sleep to conserve energy is described as a trade-off with throughput and/or latency. These are not a problem here as this protocol is designed for the traffic model described in section 6.2.1 which does not require a high throughput and can withstand delays, as long as they don't compromise the 2 minute update interval.

U-MAC has advantages over the other MAC protocols as it is specifically designed for the SmartBadge network where the crucial feature is a base station receiver node that is always going to be awake. This means that the control overhead can be reduced as all communication is to these base nodes and hence there is no need to learn other SmartBadge's sleep schedules, which in turn maximises the time spent asleep. As a result of this, U-MAC is a fairly simple protocol requiring little processing power, which will help to further reduce battery usage.

6.3 U-MAC Simulations

The NS-2 simulator [35] used for the routing protocols did not work when we tried to simulate S-MAC with multiple base stations, and it is also relatively complex to add user written models to it, so instead we used another simulator called OPNET Modeler [45] for our MAC layer simulations. This section describes the simulation model used (with the full code in Appendix C) and discusses the results collected.

6.3.1 OPNET Simulation Model

OPNET is a discrete event simulator with a graphical user interface and uses hierarchies in developing the models. At the top level is the network of nodes (an example is shown in Figure 6.11). We have used similar scenario sizes to those in the previous NS-2 simulations. They are summarised in Table 6.3.

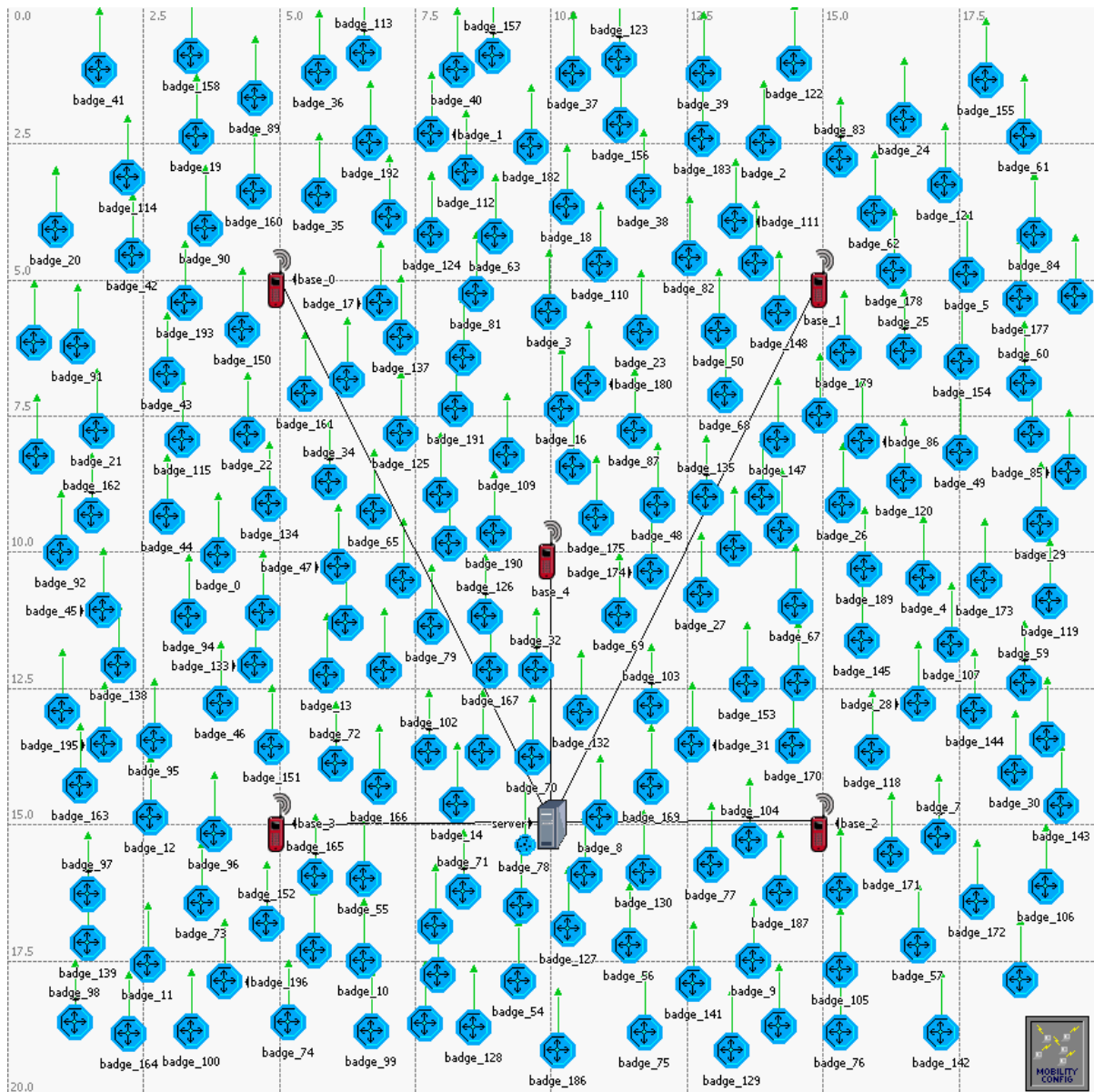


Figure 6.11 The network model for scenario 7, from OPNET [45]

In the network model you can see all the SmartBadges as mobile nodes (with random mobility vectors), and the base stations all linked to the central server. The individual objects are shown in Figure 6.12 for clarity.



Figure 6.12 The SmartBadge, base station and server objects of the network model

Scenario Number	Room Size (m)	Number of SmartBadges	Number of Base Stations	Location of Base Stations	Radio Range (m)
1	10 x 10	20	1	(5,5)	5
2	20 x 20	150	4	(5,5), (15,5), (15,15), (5,15)	5
3	20 x 20	150	4	(6,6), (14,6), (14,14), (6,14)	6
4	20 x 20	150	5	(5,5), (15,5), (15,15), (5,15), (10,10)	5
5	20 x 20	200	4	(5,5), (15,5), (15,15), (5,15)	5
6	20 x 20	200	4	(6,6), (14,6), (14,14), (6,14)	6
7	20 x 20	200	5	(5,5), (15,5), (15,15), (5,15), (10,10)	5

Table 6.3 OPNET simulation scenarios

Node mobility is handled by a “Mobility Config” object in OPNET (bottom right corner of the network model). It randomly moves all the nodes around according to the following parameters:

- A normal ‘speed’ distribution with mean 4km/hr and variance 1km/hr
- A normal ‘pause time’ distribution with mean 10 minutes, variance 3 minutes

The base station coverage areas are shown in Figure 6.13 and the calculations are in Appendix E. Part (a) is for scenario 1 with 78.54% coverage, (b) is scenarios 2 and 5 (also 78.54%), (c) is scenarios 3 and 6 (88.43%), and (d) is scenarios 4 and 7 (83.90%).

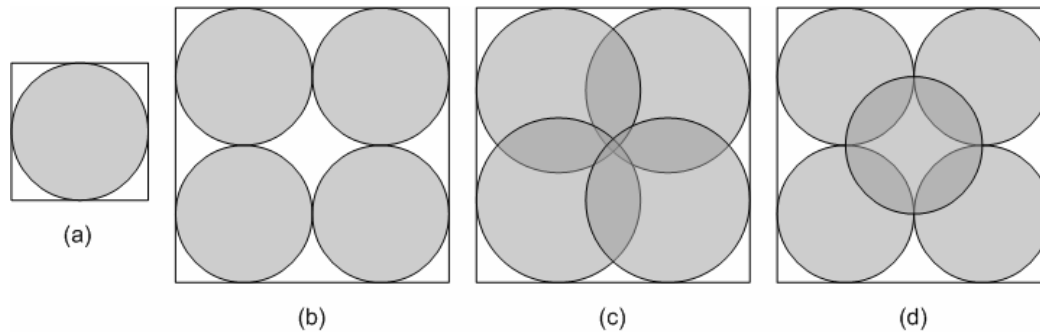


Figure 6.13 Base Station Coverage Areas

The power required for these coverage areas is determined from measurements of the prototype transmitting at minimum output power (-20 dBm) where a range of 3m was achieved. The range was defined as the point at which the receiver (sensitivity -105 dBm) could still receive data without errors; they started to appear at a distance of just over 3m in testing. As range is directly proportional to transmitted power the power

required to achieve a 5m range would be $5/3 \times 0.1\text{mW}$ (-20 dBm) = 0.167mW (-15.5 dBm).

The simulator treats this coverage boundary as a step function i.e. SmartBadges any distance over the limit (even as little as 5.0001m apart) will have no contact with each other.

6.3.1.1 SmartBadge Nodes

The level below the network model is the nodes themselves. The SmartBadge node is shown in Figure 6.14. It consists of an antenna (whose radiation pattern has been specified to match the output of SuperNEC [1]) which is connected to a radio receiver and a radio transmitter module. The receiver feeds the main processor block while packets to be sent first pass through a block which handles the carrier sense before arriving at the transmitter. The process model that handles the bulk of the work is the U-MAC protocol (described in section 6.2.2 above, with complete programme code provided in Appendix C.1) which is inside the “tx_gen_and_rx” block of the node model.

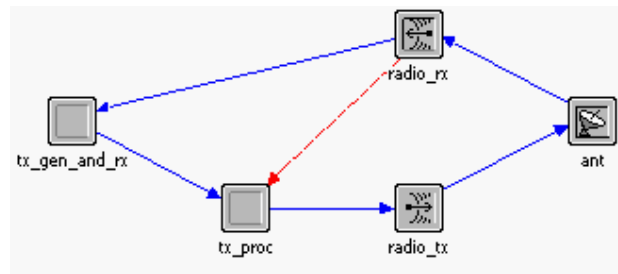


Figure 6.14 SmartBadge Node Model

SmartBadge energy is tracked by recording the time spent in different stages with the time between when a SmartBadge enters and exits the **SLEEP** state giving the idle time. Since OPNET is an event based simulator, the processing between events all happens at the same instant in time, this means the time spent in transmit is only the time taken to send the packet (16 bits @ 9600bps = 1.67ms for a control packet, 142 bits = 14.79ms for a data packet). The receive time is the remainder calculated as the time spent out of the **SLEEP** state less the transmit times. The power used in each state is the same as before i.e.

- Transmitting (5m) 82.5 mW
- Transmitting (6m) 83.7 mW
- Receiving 91.2 mW
- Idle (Sleeping) 78.0 mW

The **NO_BATT** state of Figure 6.10 is where the model goes when the energy remaining falls to zero. Nothing happens in this state; any future events simply transition back to it.

We discovered after the first run of the simulations that there were a few bit errors occurring and so sought to deal with these. The default method is to destroy any packets with errors so only error-free packets actually complete transmission (it will still report the number of bit errors occurring though these packets are never received). We re-wrote part of the channel model so that the erroneous packets would still be received and hence we could process them. If a packet has errors we send one retransmit request to try and get an error free packet, if this also has errors we give up and discard the packet as if it never arrived (like the default). Upon sending a request the SmartBadge waits in the current state (**RX_CTS** or **RX_ACK**) by transitioning along the **WAIT_RETRANS** or **ACK_RTX** loops. The retransmit request packet format is shown in Figure 6.15. Note that since the base stations do not have IDs one of these ID fields will be 0.

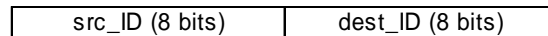


Figure 6.15 U-MAC Retransmit Packet Format

OPNET automatically knows if a packet has errors but in practice we would need to add a checksum of 1 or 2 bytes to each data packet (and possibly the control packets too) in order to detect transmission errors.

6.3.1.2 Base Station Nodes

The base station node model (Figure 6.16) is similar to the SmartBadge, except for the addition of a wired transmitter and receiver pair connected to the main processor block.

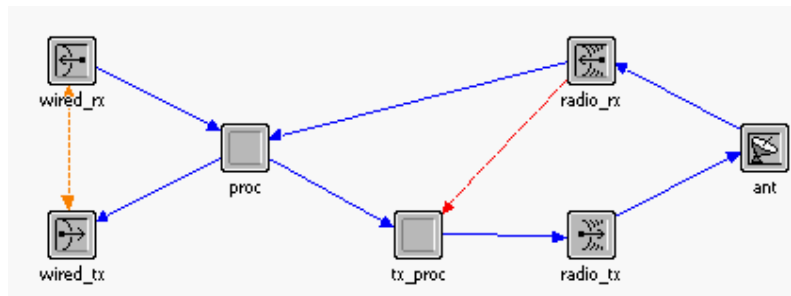


Figure 6.16 Base Station Node Model

The base stations process model is shown in Figure 6.17 and resides inside the “proc” block of Figure 6.16 (“tx_proc” is for carrier sense as in the SmartBadge model). It

only has two states, the **INIT** state and the **FORWARD** state. The **INIT** state simply registers the statistics and sets the range of the radio transceiver. The **FORWARD** state receives a new packet from the SmartBadges and forwards it to the server, or vice versa. Retransmission processing is also handled in the **FORWARD** state.

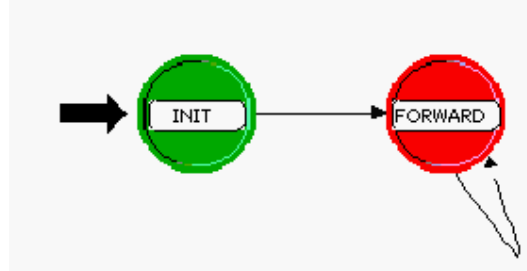


Figure 6.17 Base Station Process Model

6.3.1.3 Central Server Node

The server node model (Figure 6.18) consists of a number of wired receiver/transmitter pairs all connected to a processor block. The process model for the “server” block in the centre of the node model is shown in Figure 6.19 and has only two states. As in the other process models, the **INIT** state is used to register the statistics collected. The **REPLY** state receives an incoming RTS or DATA packet forwarded from one of the base stations. It checks the sequence number to determine if another base station has already received this packet or not. If not it sends a CTS or ACK reply respectively. If however another base station is already handling this packet then the server sends a “cancel” control packet back to the base station which lets the base station know that it should not reply to the SmartBadge.

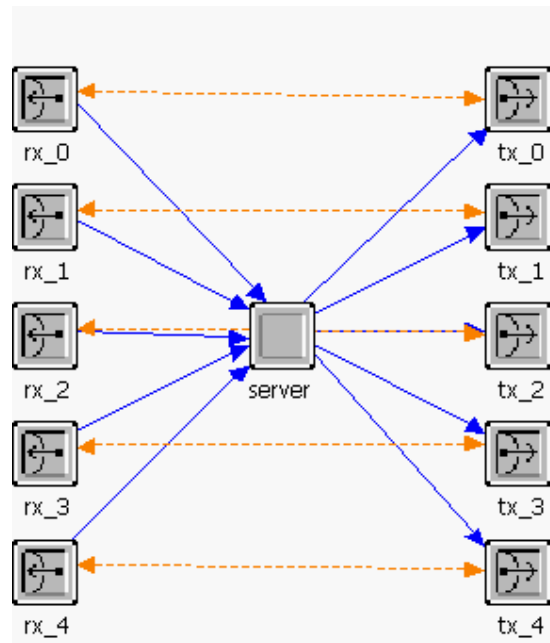


Figure 6.18 Central Server Node Model

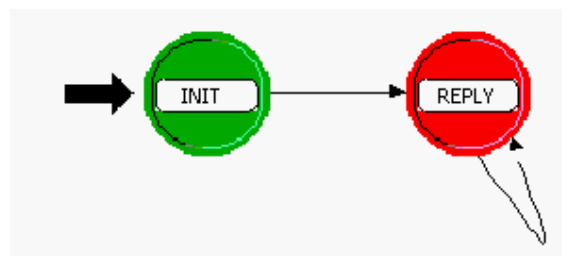


Figure 6.19 Central Server Process Model

6.3.2 Simulation Results

We have collected statistics from the OPNET simulations for the SmartBadges, the base stations and the central server, all of which are presented below.

6.3.2.1 SmartBadge Results

For the SmartBadges we collected results for the average lifetime and the number of packets passed through the entire network, as in the routing protocol simulations. We have also collected the average end-to-end delay throughout the network, and the retransmission requests sent and received.

The SmartBadges lifetime (the time at which the SmartBadges energy becomes zero) is shown in Figure 6.. The average value is just over 30500 seconds (or 8h28m20s) for all scenarios with the ranges shown by the high-low bars⁵. The average is highest in

⁵ These bars show the earliest and latest zero-energy times for individual nodes in each scenario.

scenario 1 where there is less competition for the medium, then in scenarios 4 and 7 where there is the fifth base station so there is a greater chance of finding an idle base station and hence fewer RTS packets are generated. For the routing simulations the lifetime was only 7¼ hours so by not having the SmartBadges forwarding data from others they survive over 16% longer.

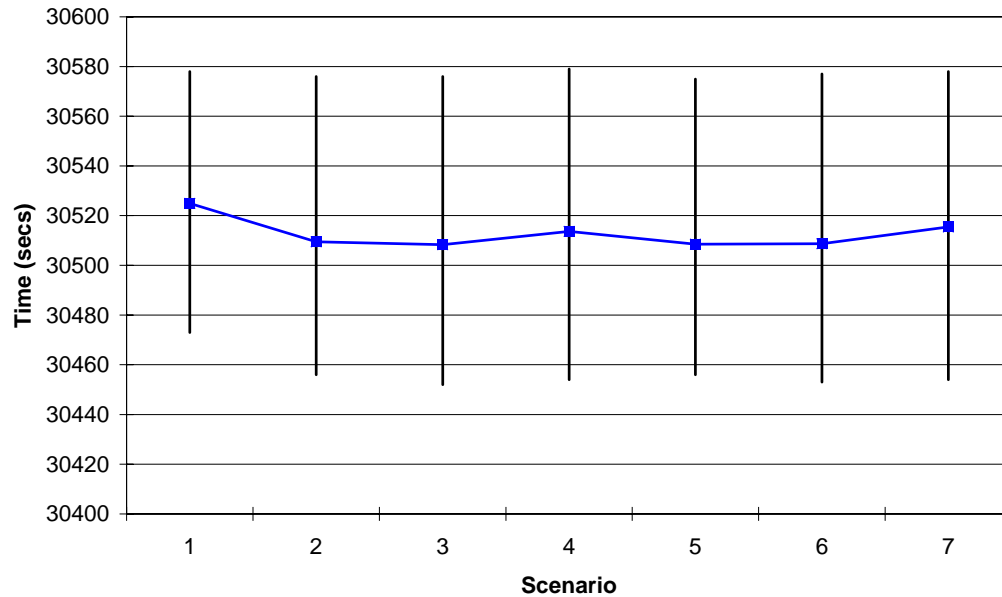


Figure 6.20 SmartBadge Lifetime

We have also recorded the percentage of time the SmartBadges spend sleeping, transmitting and receiving. This is shown in Table 6.4, where it can be seen that in all seven scenarios over 99.88% of the time is spent asleep.

	Scenario						
	1	2	3	4	5	6	7
Sleeping	99.9511%	99.9047%	99.9114%	99.9400%	99.8809%	99.9036%	99.9403%
Transmitting	0.0111%	0.0094%	0.0099%	0.0108%	0.0130%	0.0109%	0.0109%
Receiving	0.0378%	0.0859%	0.0787%	0.0492%	0.1060%	0.0855%	0.0488%

Table 6.4 Time SmartBadges spend Sleeping, Transmitting, and Receiving

The number of packets of each type sent or received by the SmartBadges is shown in Figure 6.21, with the values being normalised by dividing the total number of packets by the number of SmartBadges in each scenario. As soon as a CTS packet is received, a Data packet is generated; hence the values are the same.

There are many more RTS packets generated than CTS packets received as a result of collisions at this stage. However the ratio of data packets sent to acknowledgments received is very close to unity which shows that using the RTS/CTS exchange to reserve the medium avoids the majority of collisions of the actual data.

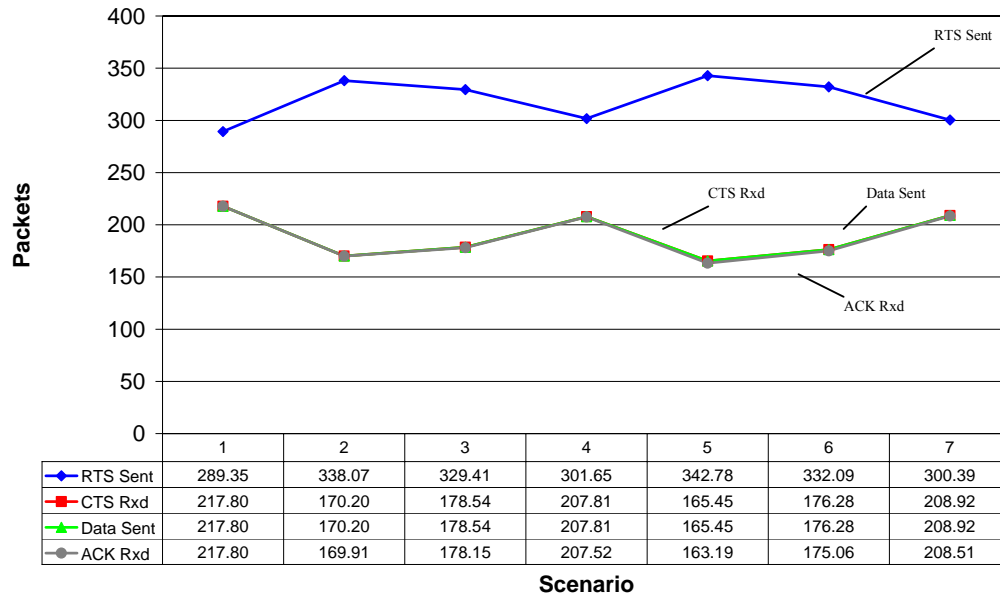


Figure 6.21 Packets Sent/Received by the SmartBadges

The end-to-end delay is plotted in Figure 6.22. For all scenarios it is close to 8ms, the variation being due to the ratio of packets sent (the scenarios with fewer data packets have a shorter delay as the main influence is the transmission time of the packet).

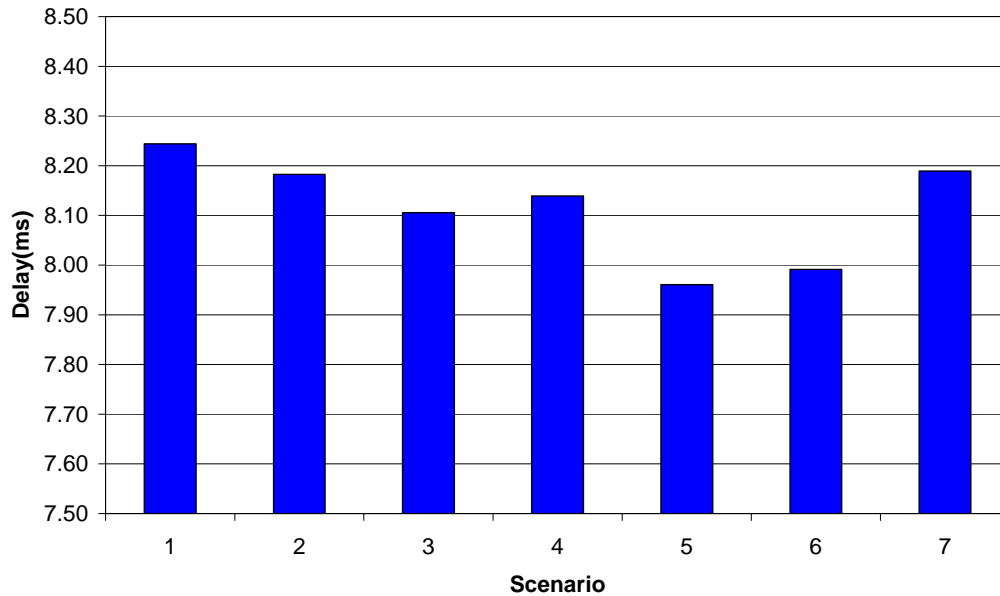


Figure 6.22 End-to-End Delay

The number of retransmit (RTX) requests sent and received by the SmartBadges is shown in Table 6.5. For most scenarios it is quite low except for scenario 5 where the number of errors suddenly skyrocketed just before the SmartBadges all ran out of battery power.

	Scenario						
	1	2	3	4	5	6	7
RTS RTX Rxd	0	1	0	0	372	42	2
CTS RTX Sent	0	1	0	0	15	4	0
Data RTX Rxd	0	1	2	1	5	2	1
ACK RTX Sent	0	3	2	0	8	3	1

Table 6.5 SmartBadge Retransmit (RTX) Requests

6.3.2.2 Base Station Results

For each base station we have recorded the number of RTS and Data packets received, as well as the number of data timeouts. We have also collected some statistics from the radio transceivers; the Bit Error Rate (BER), Signal-to-Noise Ratio (SNR), and the channel utilisation from both the receiver and transmitter.

The RTS and Data packets received by each base station are shown stacked in Figure 6.23 and Figure 6.24 respectively. Base_0 (top left) and Base_2 (bottom right) are more popular than Base_1 and Base_3 for an unknown reason (the badges are evenly spread and should move randomly so we would have expected a more even distribution). Base_4 (central) is also popular when present, but that is not surprising as users are more likely to be in the centre of an area than at the edges.

These results are not normalised, so it is expected that the number of packets received will increase both with the number of SmartBadges present and the coverage area of the base stations.

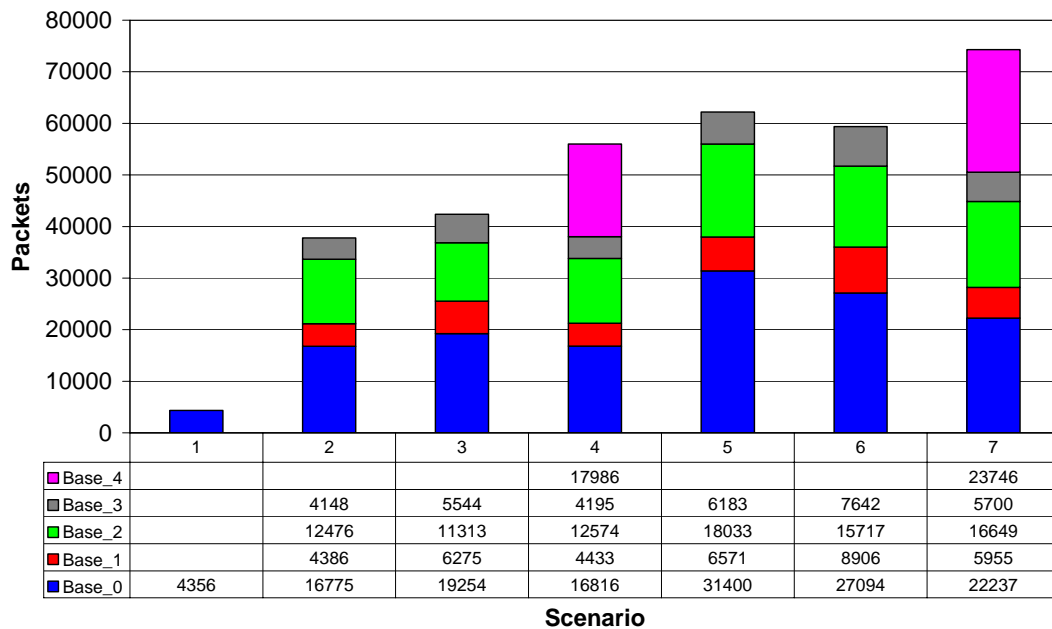


Figure 6.23 RTS Packets received by Base Stations

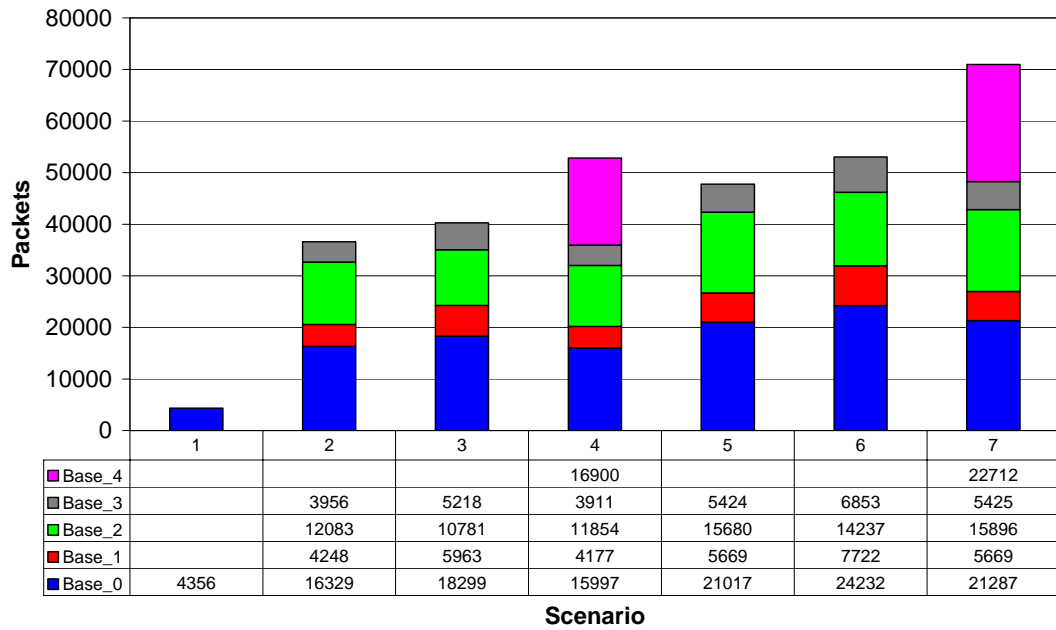


Figure 6.24 Data Packets received by Base Stations

Figure 6.25 shows the number of data timeouts at each base station. A data timeout occurs when an RTS packet is received, but the CTS presumably does not reach the SmartBadge as no data is sent from it within a reasonable time (50ms). Then a timeout occurs so that the base station is freed up to accept an RTS from another SmartBadge.

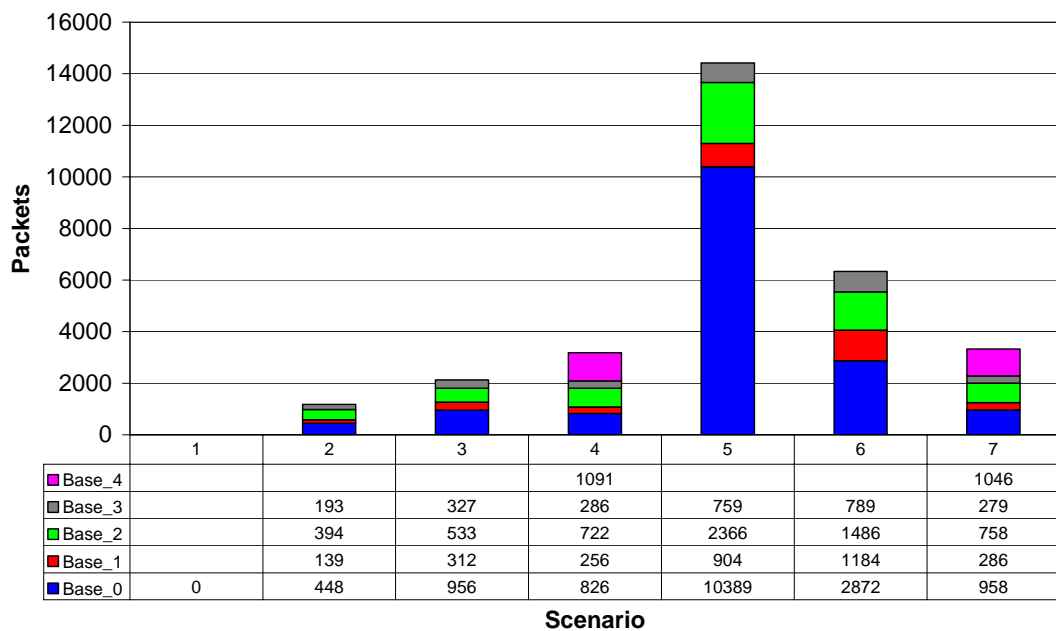


Figure 6.25 Data Timeouts for each Base Station

The spike in scenario 5 is the skyrocketing near the end of the simulation. It would appear from this that too many badges with overlapping sleep schedules made there way to base_0 and hence there was a lot of congestion while they re-scheduled themselves better, which explains an increase in retransmit packets and data timeouts (and also the BER next). This is unlikely to happen in practice as the SmartBadges are worn by people and hence cannot pack together tightly as people like their own personal space but there is no way in the simulator to limit the distance between two SmartBadges.

The next few graphs are from the built in statistics of the radio modules in OPNET. Figure 6.26 is the Bit Error Rate seen at each receiver. We discovered that by adding a random interval of up to 10ms when a SmartBadge overhears another RTS or CTS packet that the BER for all simulations dropped by a factor of at least 10, and so the BER could probably be further improved by adjusting this random interval (we also tried a maximum of 5ms which worked better for some scenarios but not others).

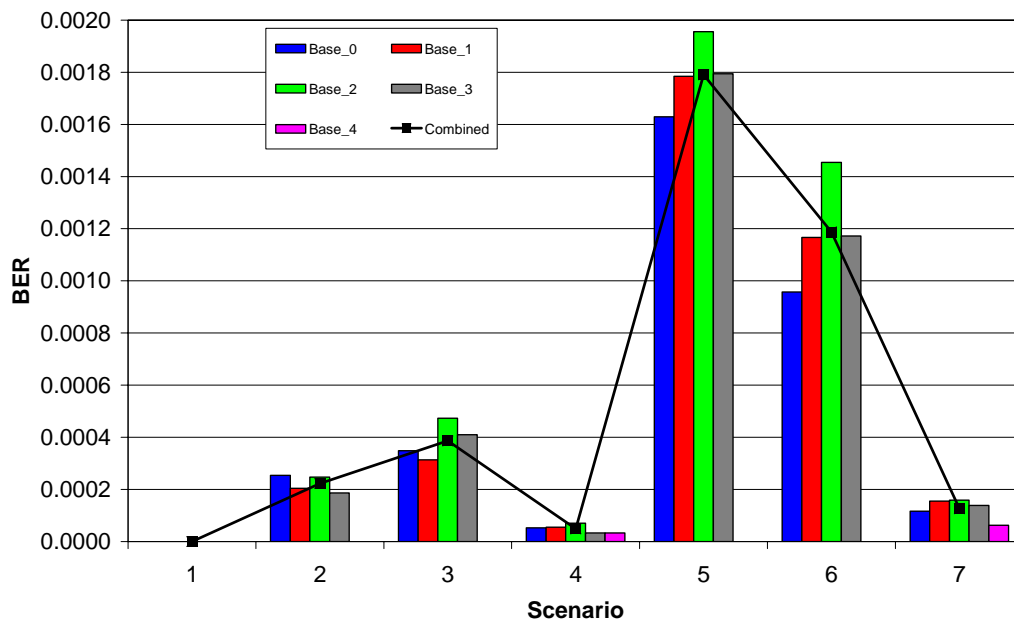


Figure 6.26 Bit Error Rate per Base Station

This is still a reasonably high BER so needs to be handled properly in practice. The simulator automatically knows if a received packet is in error but in practice we would need to add a checksum to each data packet to detect errors. If we chose to not use Manchester coding with the CC1010's RF transceiver it is possible to add some simple channel coding to the data and hence we could even correct some of the detected errors.

Figure 6.27 is the Signal-to-Noise Ratio at the receiver of each base station. Figure 6.28 is the channel utilisation (the percentage of available bandwidth, or channel capacity) at the both the receiver and transmitter. All these values (BER, SNR and Utilisation) have been averaged over the entire length of each simulation.

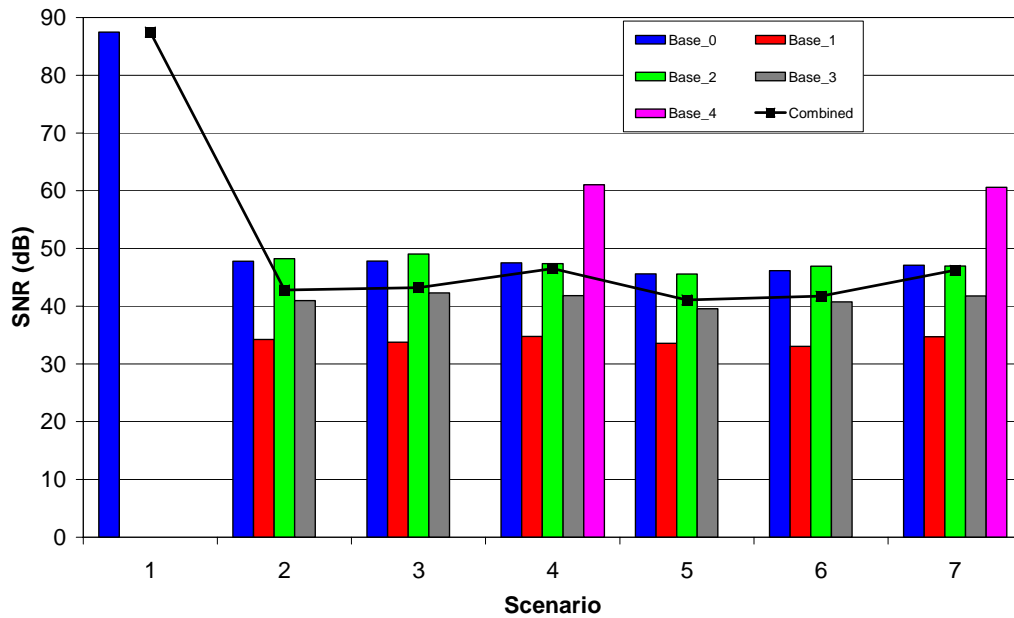


Figure 6.27 Signal-to-Noise Ratio per Base Station

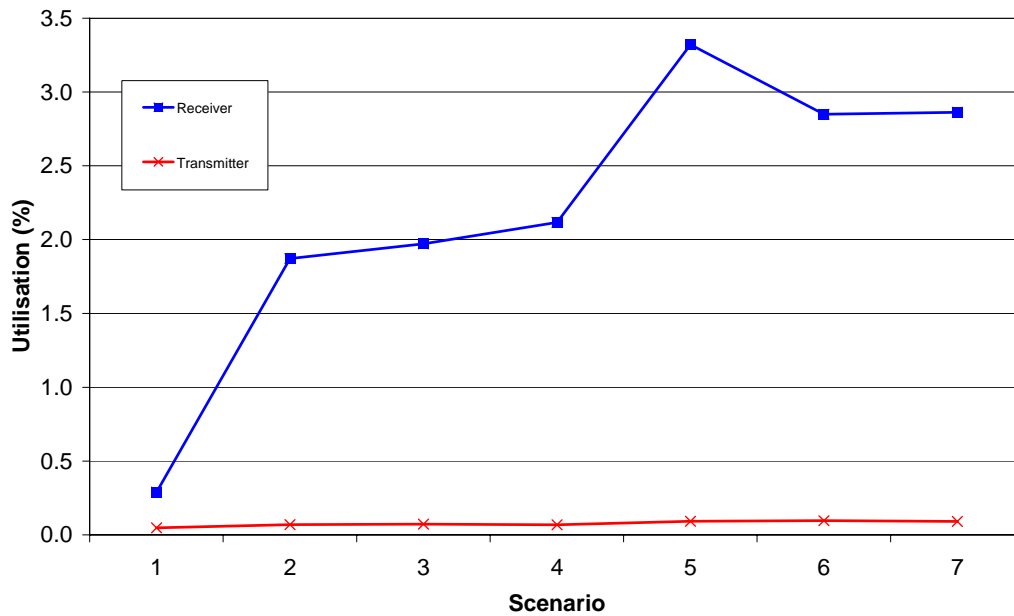


Figure 6.28 Channel Utilisation per Base Station

6.3.2.3 Central Server Results

At the central server the results obtained are the number of RTS and Data packets received, and the number of cancel packets sent back to the base stations. As you can see in Figure 6.29 most of the RTS packets received are followed by Data packets. The number of cancel packets sent gives a measure of the overhearing performed by the base stations; each packet appears to be heard by about 1.5 base stations on average.

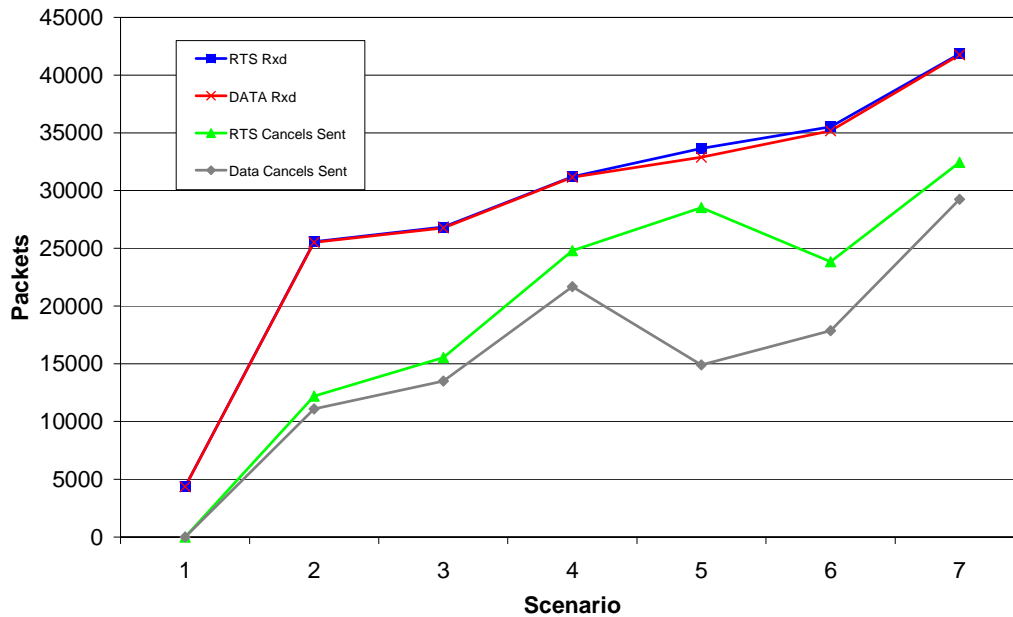


Figure 6.29 RTS, Data and Cancel packets at the Central Server

6.4 Chapter Summary

This chapter has discussed the design of the communications protocol to be used between SmartBadges and base stations. It started with simulations of two different routing protocols, DSDV [37] and AODV [40], and discovered that by using routing the SmartBadges lasted about 7¼ hours on average as some that were closer to the base station consumed a lot of energy forwarding packets from further away nodes. There were also problems with traffic congestion in the larger scenarios and so it was decided to opt for a solution with multiple base stations such that the SmartBadges saw a single hop network.

This change in approach meant that it was now a multiple access problem and we hence designed a new MAC protocol (called U-MAC) to take advantage of the SmartBadge network we had created. U-MAC exploits the knowledge that all communication is to a base station which is always awake and ready to receive. It is a

novel protocol as other protocols with a sleep cycle cater for communication between the battery powered nodes (SmartBadges in our case). By knowing that all required communications are to a mains powered node U-MAC can conserve more energy as it does not need a process to determine when the receiver will be awake. As a result of this the SmartBadges maximise the time they spend asleep (over 99.88% of the time in simulations) which has increased the lifetime by over 16% compared to the routing approach. This also compares favourably to other sleeping protocols as Ye's own experiments [15] suggest that S-MAC would sleep only about 77% of the time due to the (redundant in our case) need to learn when neighbouring nodes are active.

Chapter 7 Conclusions

This chapter first summarises all the work contained in this thesis and then provides some suggestions for future work for the SmartBadge project.

7.1 Thesis Summary

In this thesis we have designed and tested the SmartBadge hardware and have also developed communication protocols for the SmartBadge which have been verified through simulations.

7.1.1 Hardware

The hardware design involved creating an antenna structure for the RF transceiver of the CC1010. This antenna was to be constructed from PCB tracks so that it was lightweight and low profile. The chosen design was a rectangular spiral shown in Figure 3.6. A matching network for this antenna was also developed in order to get the maximum power transfer from the RF transceiver to the antenna.

In addition to the CC1010 IC and the antenna other major components of the SmartBadge are the infrared transceiver and codec, the LCD display, some LEDs, buttons and a piezoelectric buzzer. These were all progressively added to prototype designs as discussed in Appendix B. This resulted in the final SmartBadge design pictured in Figure 4.5.

7.1.2 Software

In Part II of the thesis we created communication protocol software for the SmartBadge to use to communicate with other badges and base stations.

Firstly the badge-to-badge protocol was defined in Chapter 5. This is achieved using the infrared link. The goal here was to collect another SmartBadges' ID number and a time count every 15 seconds while the two SmartBadges remain in contact with each other. The main difficulties were realising the difference in data formats expected from the CC1010 and the infrared transceiver, and also getting two SmartBadges to synchronise themselves together in order to successfully transfer data. The difference in data formats was handled with a codec (coder/decoder) chip to translate between the two formats. The synchronisation problem was solved by a form of repetition

coding where the SmartBadges waited until the same message had been received three times in a row before passing this on as being the correct message.

Having collected the data from another SmartBadge the goal of the badge-to-base protocol was to upload this information to a central server via the RF transceiver. This was finally accomplished by creating a new medium access control protocol called U-MAC (or Uplink MAC). U-MAC is a novel protocol designed to conserve as much energy as possible through the knowledge that all communication over the RF transceiver occurs between a SmartBadge and a base station that is always awake. This meant that the SmartBadge could sleep until it was ready to send the data to the server (done every two minutes) as it did not matter when neighbouring badges were awake (except for collision avoidance). This maximisation of the sleep times meant that on average the SmartBadges lasted for nearly 8½ hours on the energy of a single coin cell battery (they slept for over 99.88% of the time). Contrast this to the S-MAC experiments [15] which slept for up to 77% of the time.

7.2 Future Research

Future work for the SmartBadge project includes some changes to the hardware necessary for a mass produced version. Also there are some suggestions for future software development including some ideas for other applications.

7.2.1 Hardware

On the hardware side we have discovered that the batteries we intended to use are not suitable, but have found a solution discussed below. Chipcon has also released a new IC which has many features that would be useful in a next generation SmartBadge. Other hardware issues to be faced include finding a suitable package for the SmartBadge PCB to protect it from its user.

7.2.1.1 Battery Issues

We had intended to use a single CR2032 coin cell battery. This battery has a capacity of 220mAh, which since it operates at 3V is 2376J of energy and this was the value used in our badge-to-base simulations. We have since discovered that a coin cell can only supply a 3mA continuous current, whereas with all the peripherals we need about 30mA continuous.

We discovered a solution from a company in Newark, NY called Ultralife Batteries [46]. They make a U10004 thin cell battery [47] which operates at 3V and can supply

up to 250mA continuous current, more than sufficient for the SmartBadge. The U10004 has a capacity of 1500mAh and so using this would mean that the SmartBadges would survive for almost seven times as long as the simulations suggest.

7.2.1.2 Chipcon CC2431

Chipcon has just released their CC2431 System-on-Chip solution for 2.4GHz ZigBee/IEEE 802.15.4 with Location Engine [48]. This includes the ZigBee MAC protocol which operates at 2.4 GHz and has been designed for wireless sensor networks so the CC2431 is ultra low power. It has 128kB of flash memory (four times the CC1010) and a location engine built into the hardware which can resolve a relative location to within 0.5m over a 64m x 64m area. This would be a useful replacement for the CC1010 in a next generation SmartBadge as it would allow many more interesting applications to be developed for the SmartBadge.

7.2.1.3 Packaging

The SmartBadge is currently larger than the credit card size we had intended. The main cause of this is the LCD screen (it takes up half the space). There are smaller LCDs available but not many that operate at 3V; however if the SmartBadges went into mass production then it should be possible to get a custom built LCD of the desired size. Another option to reduce the size is to fabricate boards with more than two layers. Also, other components could be found in smaller sizes for a mass produced version which meets the goal of being the size of a credit card.

Once this size goal has been reached we would also need to consider packaging for the SmartBadge to protect the circuitry. There would need to be windows for the LCD and the IR transceiver and it would need to be able to be opened in order to replace the battery.

7.2.2 Software

Future improvements to the software could include more advanced error handling algorithms (including incorporating error correcting codes before data decisions are made) which may be necessary to support higher data rates in future applications. Any such modifications to the software need to consider how they effect the energy consumption of the SmartBadge as the current amount of data transferred is minimal and any increase in control overhead may not be worth the energy cost.

7.2.2.1 Applications

The SmartBadges have been developed based on the conference application. This is where they are used primarily to replace a business card as they seamlessly collect contact details from people that you interact with, which can then be ordered by importance based on the contact time.

Other applications that have been suggested include:

- Embedding SmartBadges in display booths at conferences so that companies could form a list of potential customers.
- Using SmartBadges in a museum setting, the ID of the user would be transferred to the exhibit and their details could be incorporated into an interactive experience.
- By storing a users interests along with their contact details the SmartBadges could make inferences about the kind of people a user is interested in talking to. If this is combined with a location engine (e.g. as in the CC2431) then the SmartBadge could be used to guide a user to another yet unmet person with similar interests to those that the user has already talked with. This would be a useful augmentation to any social networking event.
- Icebreaker activities like randomly pairing two users together could be enhanced by keeping track of whom someone has walked past and hence the SmartBadge could guide a user towards their secret partner. The match could then be signified using the buzzer. It could also be recorded and uploaded in real time to award a prize to the first pairs to find each other.

Applications that would most benefit from the SmartBadge hardware are those that require data from the whole group to be collected in real time. For example a poll could be conducted with the question displayed on the LCD, using the two buttons as accept and reject, and the results sent through the RF network almost instantaneously.

Glossary

ABR	Associativity-Based Routing
ACK	Acknowledgment, a packet sent from the receiver back to the transmitter to indicate successful reception
ADC	Analogue to Digital Converter
AODV	Ad-hoc On-demand Distance Vector Routing
ASCII	American Standard Code for Information Interchange; a common method used to represent letters using an 8-bit binary sequence
BER	Bit Error Rate
BFSK	Binary Frequency Shift Keying
BNC	Bayonet Neill Concelman, a common connector for co-axial cables
CBR	Constant Bit Rate
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Multiple Access
CSMAC	CDMA Sensor Medium Access Control
CGSR	Clusterhead Gateway Switch Routing
CTS	Clear to Send; a receiving node sends this packet in response to an RTS so that others within range of receiver and not the transmitter will not interfere with the message (hidden terminal problem)
DES	Data Encryption Standard
DMAC	Data-gathering Medium Access Control
DS	Data Send, see T-MAC (2.3.2.1)
DSDV	Destination Sequenced Distance Vector Routing
DSMAC	Dynamic Sensor Medium Access Control
DSR	Dynamic Source Routing

FRTS	Future Request To Send, see T-MAC (2.3.2.1)
HITLabNZ	Human Interface Technology Laboratory New Zealand; located at the University of Canterbury this is where the research was performed
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronic Engineers
IF	Intermediate Frequency
I/O	Input / Output
IR	Infrared (radiation at wavelengths between 700nm and 1mm); the infrared device in the SmartBadge has a peak wavelength at 890nm
IrDA	Infrared Data Association
ISM	Industrial, Scientific, Medical; the ISM RF bands do not require a license to operate in and so are used by many wireless networking devices (868 MHz is in an ISM band, so is 2.4 GHz, the typical IEEE 802.11 frequency)
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least Significant Bit
MAC	Medium Access Control; a means of deciding which device can use the medium (radio link) at which point in time
MATLAB	Matrix Laboratory
MHz	Megahertz; 1,000,000 Hz
MS-MAC	Mobility aware Sensor Medium Access Control
NFC	Near Field Communication
NRZ	Non Return to Zero, i.e. no coding. A '1' bit sent as a high frequency without the low frequency following and a '0' bit is just the low frequency
NS-2	Network Simulator 2
PAN	Personal Area Network

PCB	Printed Circuit Board
PMAC	Pattern Medium Access Control
PWM	Pulse Width Modulator
RF	Radio Frequency, frequencies between 3 Hz and 300GHz; the CC1010 can operate at frequencies between 300 and 1000 MHz however it is optimised for the ISM bands like the 868 MHz that we use
RFID	Radio Frequency Identification
RS232	Recommended Standard 232; a common protocol for serial ports on computers
RSSI	Received Signal Strength Indicator
RTC	Real Time Clock
RTS	Request to Send; a transmitting node sends this packet and waits for a CTS reply so that others within range of the transmitter and not the receiver will not interfere with the message (hidden terminal problem)
RX	Receive
SIR	Serial Infrared
S-MAC	Sensor Medium Access Control
SNR	Signal-to-Noise Ratio
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSR	Signal Stability Routing
SuperNEC	Super Numerical Electromagnetic Code
TDMA	Time Division Multiple Access; a form of MAC where the access is divided up into time slots and users transmit during their allocated slot only
T-MAC	Timeout Medium Access Control
TX	Transmit
UART	Universal Asynchronous Receive Transmit

UDP	User Datagram Protocol
U-MAC	Uplink MAC; the MAC protocol designed for the SmartBadges
URL	Uniform Resource Locator, i.e. a World Wide Web address
VC	Virtual Cluster, a collection of nodes with the same sleep/wakeup schedules
WiseMAC	Wireless Sensor Medium Access Control
WRP	Wireless Routing Protocol
ZigBee	a new MAC protocol developed for wireless sensor networks, also called IEEE 802.15.4

Appendix A Hardware Details

This appendix gives the details of the hardware platform for the SmartBadge. It includes circuit details for each of the peripherals as well as the overall schematics and the PCB layout.

A.1 Peripherals

In this section we present the circuits for each of the peripherals used and describe how they function. Also included are references to the relevant datasheets.

A.1.1 Infrared (IR) Transceiver

The infrared transceiver is a TDFU4100 [34] from Vishay Electronic shown in Figure A.1 below. It has to be connected to the CC1010 via an interface IC, the TOIM4232 codec [33] (Encoder / Decoder) also by Vishay, as it expects signals which don't match those output by the UART module on the CC1010.

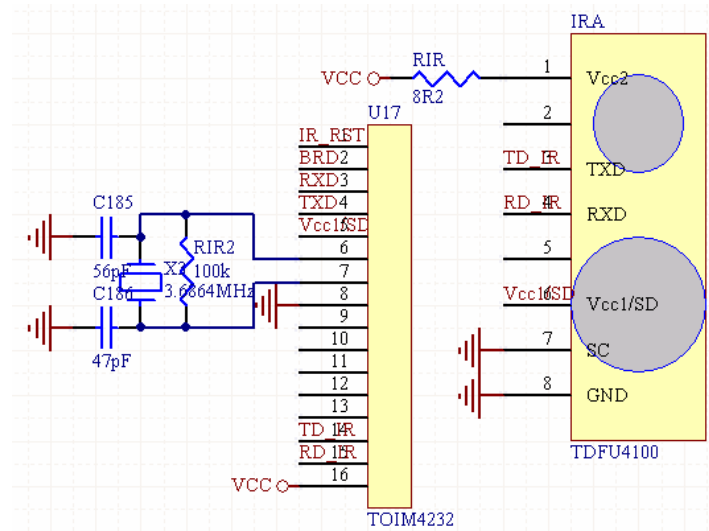


Figure A.1 Infrared Circuit Diagram

The signals RXD and TXD on pins 3 and 4 of the TOIM4232 are the receive and transmit lines on the CC1010 UART, the data passes out of the TOIM4232 on pins 14 and 15 which feed into the TDFU4100. The IR_RST is a control line from the CC1010 which returns the TOIM4232 to its default baud rate, the Vcc1/SD output that connects to the TDFU4100 is the inverse of this reset signal and controls the power for the infrared LED. The other control line (BRD) is used to toggle between

programming the baud rate on the TOIM4232 and sending data to it to pass on to the TDFU4100.

The TOIM4232s baud rate generation is done using the 3.6864 MHz crystal attached to pins 6 and 7. The 8.2Ω resistor on the Vcc input of the TDFU4100 is to limit the range of the infrared LED; it is the recommended value from the datasheet.

A.1.2 Liquid Crystal Display (LCD)

Figure A.2 below is the circuitry for the LCD module; it consists of the LCD [49] itself and a 74HC164 [50] 8-bit serial in parallel out shift register. The shift register gets 8-bit words send from the CC1010s SPI interface on the MOSI pin with SCK being the clock for timing. !CLR is used to reset the shift registers output to zero. This data is read into the LCD module when the enable (E) pin is set high, RS is the register select to decide whether the data is an instruction or display data. A and K are the anode and cathode of the backlight which we are not using (to conserve power), V0 is used to control the contrast of the display, tying it to ground means full contrast (and hence less need for a backlight). The final input is R/W (read/write) which can be used to read data out of the instruction and data registers in the LCD module, but we decided we did not need to be able to read the data out again so it is set low to permanently treat the DB0-DB7 lines as incoming data.

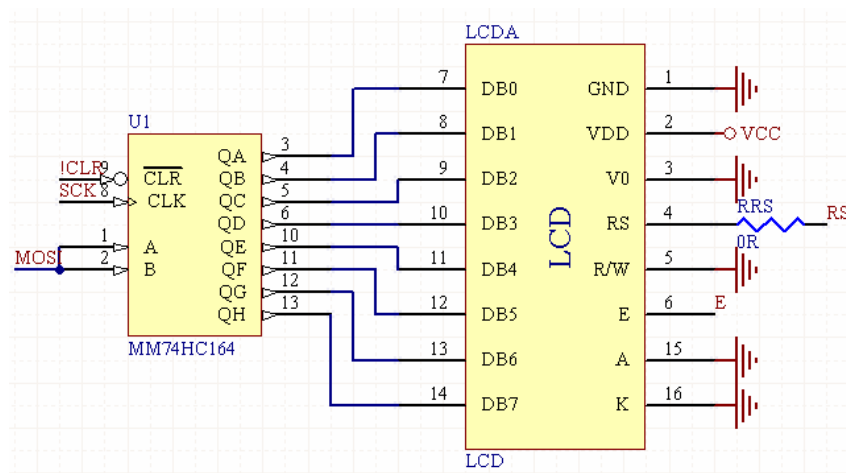


Figure A.2 LCD Circuit Diagram

A.1.3 Piezoelectric Buzzer

The buzzer [51] shown in Figure A.3 is connected to the CC1010s PWM output via a $2k2\Omega$ resistor. The PWM outputs a 50% duty cycle square wave at about 4 kHz as that frequency gives the maximum volume from the buzzer.

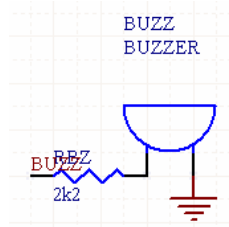


Figure A.3 Buzzer Circuit Diagram

A.1.4 Light Emitting Diodes (LEDs)

There are four bi-colour (red/green) LEDs [52] on the SmartBadge shown in Figure A.4; each requires two I/O lines from the CC1010. They consist of two LEDs in a single package (one red one green) so one can light them either green or red or both (yellow). The LEDs are connected via a 100Ω resistor to Vcc and so setting the relevant pin (denoted LEDXY where X is 1-4 and Y is A or B) to logic ‘0’ will turn the LEDs on. The resistors are all in a single networked package to save space.

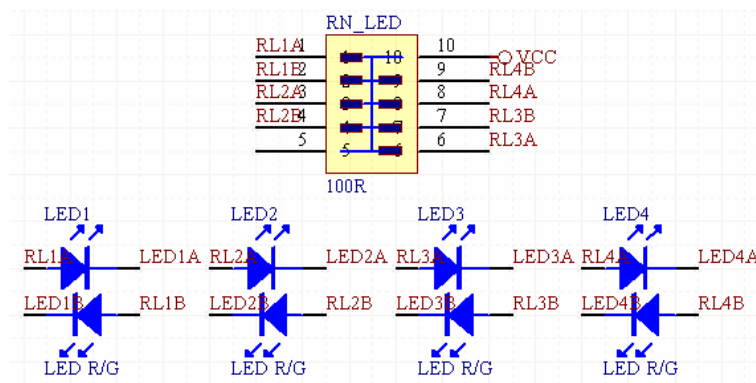


Figure A.4 LED Circuit Diagram

A.1.5 Pushbuttons

The two buttons [53] on the SmartBadge are connected between the CC1010 external interrupt inputs and ground (see Figure A.5). They also have a 47kΩ resistor to Vcc. In this way the input value is logic ‘1’ when the buttons are open and goes to logic ‘0’ when they are pressed.

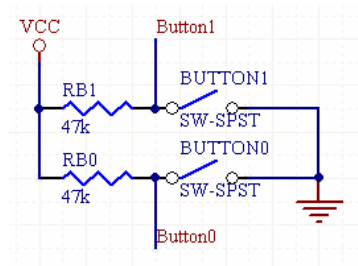


Figure A.5 Button Circuit Diagram

A.2 Schematic

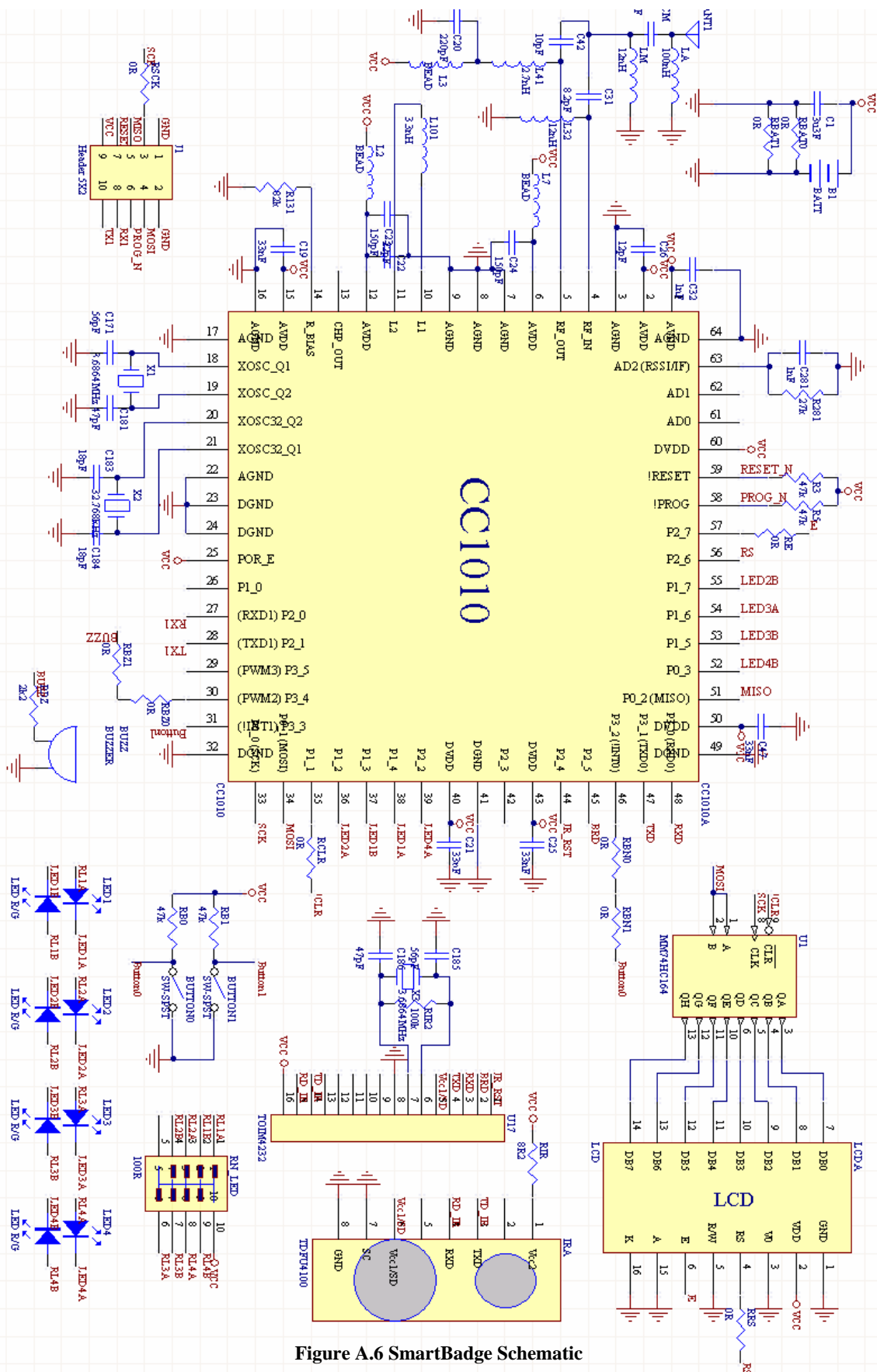
The final circuit schematic for the SmartBadge is shown in Figure A.6 on the next page. Other than all the peripherals mentioned in the previous section things to note include:

- The CC1010 runs from two crystals. The main oscillator is running at 3.6864 MHz and the RTC oscillator is at 32.768 kHz. By switching to this slower crystal it is possible to dramatically reduce the CC1010 power consumption so this is done when in idle mode.
- Pin 63 has a resistor and capacitor connected to it so that measurements of the RSSI of the RF receive signal can be made.
- There are quite a few filtering capacitors between various Vcc and Ground pins as recommend in the CC1010 Datasheet [32].
- The components connecting to the antenna are again Chipcon recommendations except for the last three (LM, CM and LA) which are the matching network described in 3.4.1.1.

A.3 PCB Layout

The PCB layout is shown in Figure A.7 and Figure A.8 on the next couple of pages. It is a two layer board as we were unable to produce more layers which meant that there are quite a few vias used in the routing process. The large hole up the top is for the LCD module; we cut this hole out of the PCB and squeezed the LCD in so as to reduce the profile of the SmartBadge. The ground plane covers most of the rest of the design on both sides except for underneath the antenna as our simulations had shown the antenna has a better gain without the ground plane present.

All the peripherals appear on the top side of the SmartBadge while the CC1010 and the battery are on the reverse. Also on the reverse is a 10 pin header used to program the CC1010.



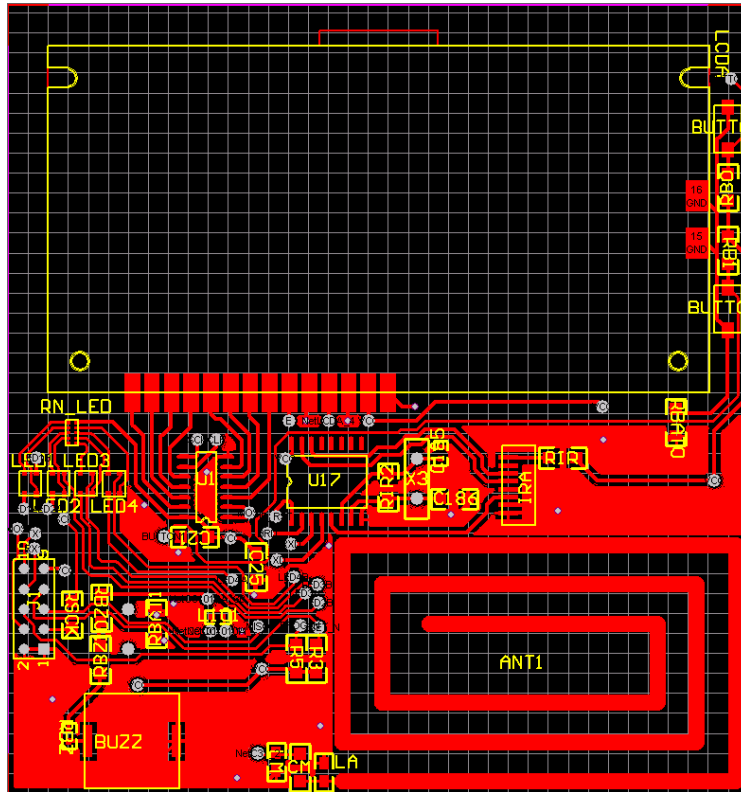


Figure A.7 PCB Layout Top Side

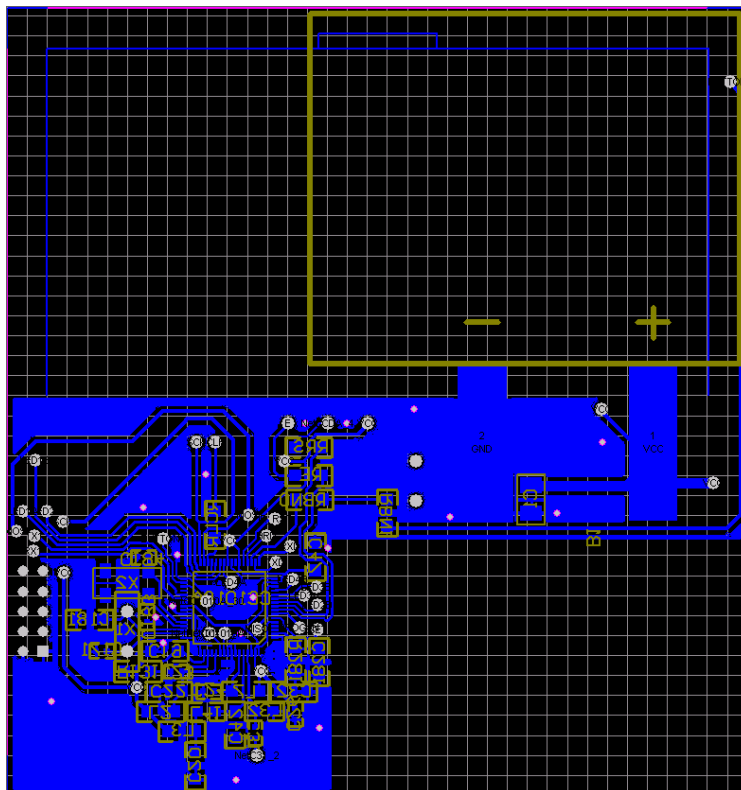


Figure A.8 PCB Layout Bottom Side

Appendix B SmartBadge Prototypes

This appendix summarises the prototypes constructed for the SmartBadge hardware platform and hence gives insight into the design process that has been taken.

B.1 First Prototype

The first prototype was vastly different from the final SmartBadge design. The goal of the first prototype was to test the original circular loop antenna design and the RF transceiver. We also had a number of pins available to connect peripherals from a breadboard which at this stage were just the LEDs, the LCD screen and buttons. The prototype is pictured in Figure B.1 below.

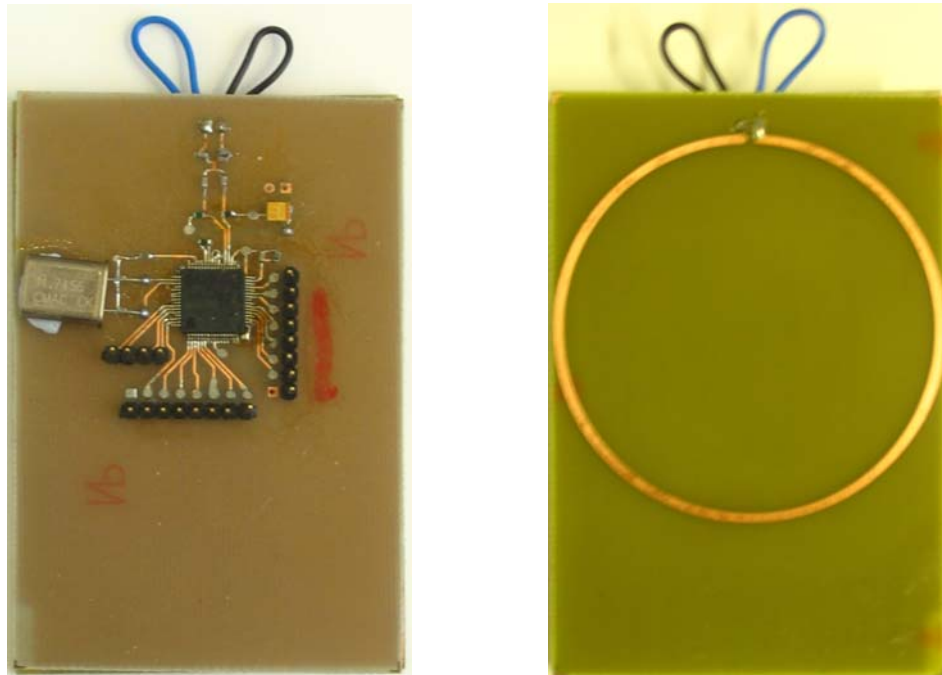


Figure B.1 The first prototype, front and back

The antenna is 6 cm in diameter and is the reason for the large size of the boards compared to the circuitry. The board measures 60 x 90 mm with the idea being that the LCD screen would fit in the extra space next to the antenna in the final design.

B.2 Second Prototype

The second prototype (pictured in Figure B.2) was eventually a success. It again did not have all the peripherals attached as we had already proven the LCD can work (using a screen with the same controller as the desired LCD), LEDs and buttons are

trivial and the buzzer was run from the PWM so was also very straightforward. The board measures 65mm high by 60mm wide.

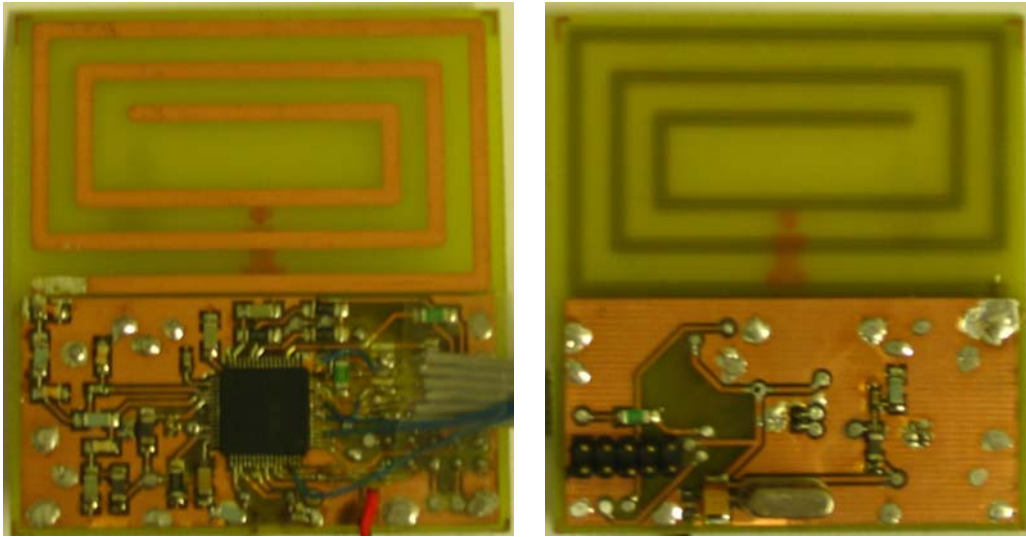


Figure B.2 The second prototype design, front and back

By this time the specifications had been changed so we had a new antenna design to verify and also the infrared circuit to test. Since we did not have another infrared device available this meant that we built two of these prototypes.

The RF section did not initially communicate with the development kit which we narrowed to a hardware problem. We set up an RF test set to see if the prototype was sending anything at all and discovered that the peak was not at 868 MHz but actually about 250 kHz below that and hence it could communicate with the other prototype but not the development kit. The reason for this turned out to be the load capacitance on the main oscillator crystal as we had not been able to source the right frequency crystal with the same capacitance as the CC1010 datasheet but had not realised this when using the example design. We adjusted the load capacitance to suit and the second prototype now communicated fine with the development kit boards. There appears to be no other adverse effects of changing this load capacitance.

As discussed in section E.2 we had to add a codec⁶ (coder/decoder) for the infrared but this could not fit on the original board so we made a small extension board shown in Figure B.3 below (which also had a buzzer) and with this extension the infrared now worked as expected.

⁶ The TOIM4232 datasheet [33] refers to it as an endec (encoder / decoder)

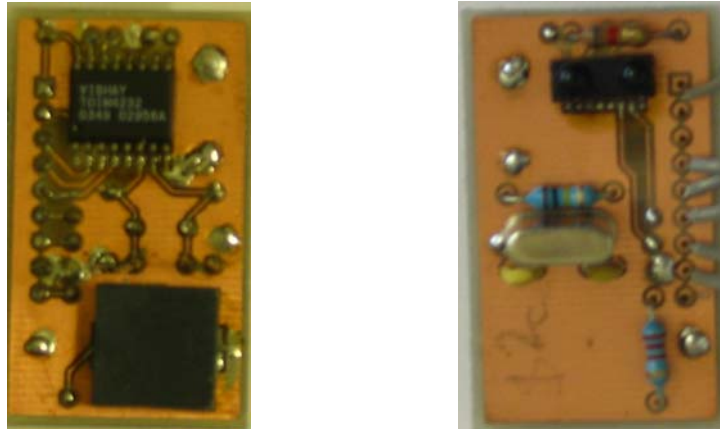


Figure B.3 The extension to the second prototype necessary for the infrared transceiver

B.3 Final Design

The final look of the SmartBadge is pictured in Figure B.4 below. It measures about 10 by 10 cm. The larger than desired size is due to the LCD screen as we were not able to find a smaller one operating at 3.3V. This design has all the peripherals (LCD, 4 LEDs, buzzer and buttons) as well as the infrared transceiver and the RF antenna on the front and the CC1010 on the rear. We did have a slight problem with the battery not being able to deliver enough current but found a solution to this as is discussed in section G.2.1.1.

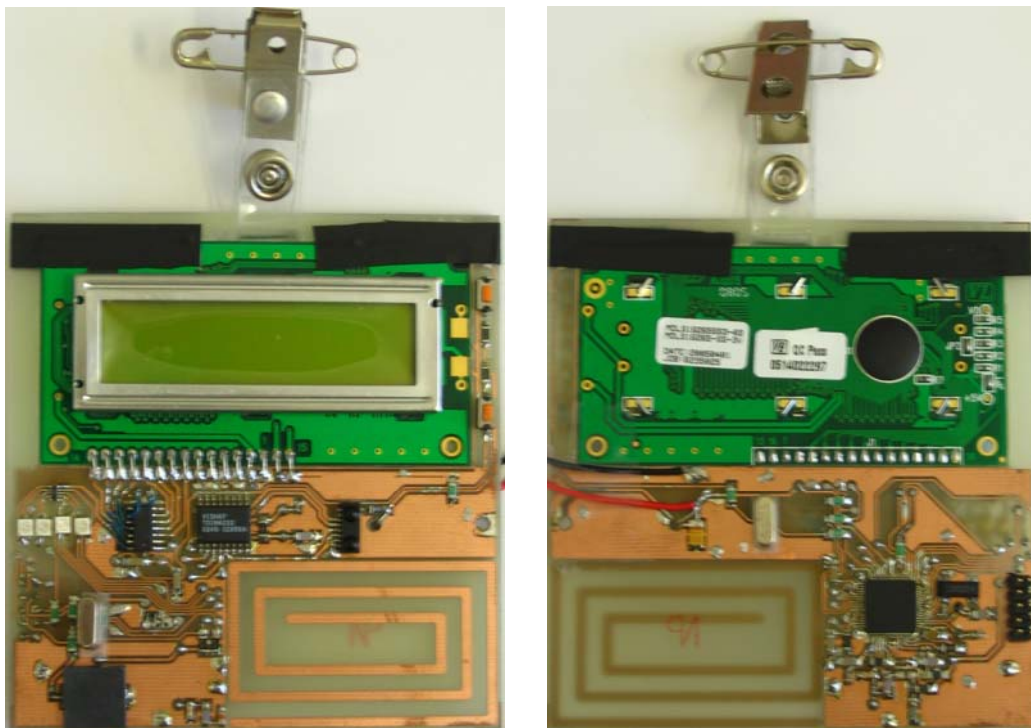


Figure B.4 The final SmartBadge design, front and back

Appendix C Routing Protocols

This appendix gives an overview of both DSDV and AODV. It also presents the simulations script written for the routing protocol simulations done in NS-2.

C.1 Destination Sequenced Distance Vector Routing (DSDV)

Destination Sequenced Distance Vector (DSDV) routing was developed by Perkins and Bhagwat in 1994 [37] and is an example of a table driven routing protocol. This means that each node maintains a table of routing information and hence knows exactly how to get from any node to any other at all times. In DSDV every node has a table containing all possible destinations and the number of hops required to reach them. These entries are marked with a sequence number assigned by the destination node, used to distinguish new and old routes.

Route updates are done two different ways; the first is a full dump which has all available routing information. The other way is by incremental packets which only contain the changes since the last full dump; this is stored in a separate table. New route broadcasts contain the destination, hop count, sequence number of the information received and a sequence number unique to the broadcast. This broadcast sequence number is used so that the most recent routes are selected. Should two routes have the same broadcast sequence number the route with the smallest metric is chosen to optimise the path. DSDV is inefficient because of the requirement for periodic update transmissions regardless of any route changes.

C.2 Ad-Hoc On-Demand Distance Vector Routing (AODV)

Source initiated on demand routing creates routes only when desired by the source. When a node requires a route it begins a route discovery process throughout the network, this route is maintained by a maintenance procedure until the destination becomes unreachable or the route is no longer needed. Ad-hoc On-demand Distance Vector (AODV) routing is source initiated and builds on DSDV; it was developed by Perkins and Royer in 1999 [40]. It is an improvement over DSDV as it minimises the number of routing broadcasts by sending them on demand only, also nodes not in the path do not maintain routing information for that path.

To find a route the source node broadcasts a route request (RREQ) packet which is forwarded until either the destination is reached or an intermediate node with a fresh enough route is located. AODV still uses destination sequence numbers to ensure routes are loop free and up to date.

Each node maintains its own sequence number and a broadcast ID. This ID is incremented every time it sends out a request such that the ID and the nodes address uniquely identify a broadcast. The RREQ request contains this info as well as the most recent destination sequence number the node has and intermediate nodes only reply if they have a destination sequence number greater than or equal to that contained in the request.

When forwarding a request a node keeps track of the first node from which it received the request in order to establish a reverse path. Once the route is found a route reply (RREP) message is sent back through this reverse path and when a node receives the reply it adds the downstream node to its routing table, hence any AODV node only knows its immediate up and downstream neighbours in any route.

Route maintenance is done as follows. If a source node moves it just sends another route request broadcast. If, however, an intermediate node moves, its upstream neighbour sends a link failure message back up to the source node via all the other upstream nodes. The source then decides whether or not to find a new route.

C.3 NS-2 Simulation Script

```
#=====
# Define options
#=====

set val(chan)          Channel/WirelessChannel
set val(prop)          Propagation/TwoRayGround
set val(netif)         Phy/WirelessPhy
set val(mac)           Mac/802_11
set val(ifq)           Queue/DropTail/PriQueue
set val(ll)            LL
set val(ant)           Antenna/OmniAntenna
set val(x)             10      ;# X dimension of the topography
set val(y)             10      ;# Y dimension of the topography
set val(ifqlen)        50      ;# max packet in ifq
set val(seed)          20.0
set val(adhocRouting)  AODV
set val(nn)            21      ;# how many nodes are simulated
set nn                 [expr ($val(nn) - 1)]
set val(start)         0.0
set val(stop)          600.0 ;# simulation time
set val(range)         3
set val(pkt)           5      ;#time between packets
```

```

set num                                "one"

##take in command line args if given
if {$argc > 1} {
    set val(adhocRouting)    [lindex $argv 0]
    set val(range)           [lindex $argv 1]
    set val(x)                [lindex $argv 2]
    set val(y)                [lindex $argv 3]
    set nn                    [lindex $argv 4]
    set num                   [lindex $argv 5]
    set val(nn)               [expr $nn + 1]
}

set val(sc)                          "../Scen-$val(x)x$val(y)-$nn"

puts "Protocol: $val(adhocRouting)"
puts "Range: $val(range)m"
puts "Scenario: $val(sc)"

=====
# Main Program
=====

# create simulator instance
set ns_                               [new Simulator]

# setup topography object
set topo                              [new Topography]

# create trace object for ns and nam
set tracefd [open $num-$val(adhocRouting)-$val(x)x$val(y)-$nn-
$val(range).tr w]
#set namtrace      [open $num-$val(adhocRouting)-$val(x)x$val(y)-$nn-
$val(range).nam w]

$nns_ trace-all $tracefd

# define topology
$topo load_flatgrid $val(x) $val(y)

# Create God
set god_ [create-god $val(nn)]

#set range
if {$val(range) == 3} {
    $val(netif) set RXThresh_ 2.13643e-05      ;#3m
    set val(tx) 4.878 ;#-20dBm
} elseif {$val(range) == 5} {
    $val(netif) set RXThresh_ 7.69113e-06      ;#5m
    set val(tx) 4.95 ;#-15dBm
}

#set data rate
if {$val(mac) == "Mac/802_11"} {
    $val(mac) set datarate_ 12.48Mb
}

#create channel
set chan [new $val(chan)]

```

```

#global node setting
$ns_ node-config -adhocRouting $val(adhocRouting) \
    -llType $val(ll) \
    -macType $val(mac) \
    -ifqType $val(ifq) \
    -ifqLen $val(ifqlen) \
    -antType $val(ant) \
    -propType $val(prop) \
    -phyType $val(netif) \
    -channel $chan \
    -topoInstance $topo \
    -energyModel EnergyModel \
    -rxPower 5.472 \
    -txPower $val(tx) \
    -idlePower 4.68 \
    -initialEnergy 2376 \
    -agentTrace ON \
    -routerTrace OFF \
    -macTrace OFF \
    -movementTrace ON

# Create the specified number of nodes [$val(nn)] and "attach" them
# to the channel.
for {set i 0} {$i < $nn} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0           ;# enable random motion
}

#Create BASE STATION in centre of room
$ns_ node-config -initialEnergy 10e10
set node_BS [$ns_ node]
$node_BS random-motion 0
$node_BS set X_ [expr $val(x) / 2.0]
$node_BS set Y_ [expr $val(y) / 2.0]
$node_BS set Z_ 0.0

# Define node movement model
puts "Loading scenario file..."
source $val(sc)

if {$num=="one"} {
    set udp_(0) [new Agent/UDP]
    $ns_ attach-agent $node_(0) $udp_(0)
    set null_(0) [new Agent/Null]
    $ns_ attach-agent $node_BS $null_(0)
    $ns_ connect $udp_(0) $null_(0)
    set cbr_(0) [new Application/Traffic/CBR]
    $cbr_(0) set packetSize_ 10
    $cbr_(0) set interval_ 5.0
    $cbr_(0) set random_ 1
    $cbr_(0) set maxpkts_ 500
    $cbr_(0) attach-agent $udp_(0)
    $ns_ at $val(start) "$cbr_(0) start"
} elseif {$num=="all"} {
    #tcp between all and BS
    for {set i 0} {$i < $nn} {incr i} {
        set udp_($i) [new Agent/UDP]
        $ns_ attach-agent $node_($i) $udp_($i)
        set null_($i) [new Agent/Null]
        $ns_ attach-agent $node_BS $null_($i)
    }
}

```

```

        $ns_ connect $udp_($i) $null_($i)
        set cbr_($i) [new Application/Traffic/CBR]
        $cbr_($i) set packetSize_ 10
        $cbr_($i) set interval_ 5.0
        $cbr_($i) set random_ 1
        $cbr_($i) set maxpkts_ 500
        $cbr_($i) attach-agent $udp_($i)
        $ns_ at $val(start) "$cbr_($i) start"
    }
}

# Define node initial position in nam
for {set i 0} {$i < $nn} {incr i} {
    $ns_ initial_node_pos $node_($i) 2
}
$ns_ initial_node_pos $node_BS 2

# Tell nodes when the simulation ends
for {set i 0} {$i < $nn} {incr i} {
    $ns_ at $val(stop).0 "$node_($i) reset";
}
$ns_ at $val(stop).0 "$node_BS reset";

$ns_ at $val(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"

puts $tracefd "M 0.0 nn $nn x $val(x) y $val(y) rp
$val(adhocRouting)"
###puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

puts "Starting Simulation..."
$ns_ run

```


Appendix D OPNET Source Code

This appendix covers all the source code written for the U-MAC protocol simulations in OPNET. It also has the derivations of the coverage areas of the base stations.

D.1 SmartBadge

For the SmartBadge nodes code was written into the Enter Executives block for the INIT, SLEEP and RX_AXK states and into the Exit Executives blocks of the INIT, SLEEP, RX_CTS and RX_ACK states. We also wrote into the Header and Function Blocks.

D.1.1 INIT Enter Executives

```
/* Initilaize the statistic handles */
threshold_lsh = op_stat_reg ("Rx Power (dBm)",
    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
//packet stats
RTSs_sent_lsh = op_stat_reg ("Pkts.RTS Pkts Sent",
    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
CTS_lsh = op_stat_reg ("Pkts.CTS Pkts Rxd", OPC_STAT_INDEX_NONE,
    OPC_STAT_LOCAL);
pkts_sent_lsh = op_stat_reg ("Pkts.Data Pkts Sent",
    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
ACK_lsh = op_stat_reg ("Pkts.ACKs Rxd", OPC_STAT_INDEX_NONE,
    OPC_STAT_LOCAL);
drop_lsh = op_stat_reg ("Pkts.Data Pkts Lost", OPC_STAT_INDEX_NONE,
    OPC_STAT_LOCAL);
cts_rtx_lsh = op_stat_reg ("RTX.CTS RTX Sent", OPC_STAT_INDEX_NONE,
    OPC_STAT_LOCAL);
ack_rtx_lsh = op_stat_reg ("RTX.ACK RTX Sent", OPC_STAT_INDEX_NONE,
    OPC_STAT_LOCAL);
rts_rtx_lsh = op_stat_reg ("RTX.RTS RTX Rxd", OPC_STAT_INDEX_NONE,
    OPC_STAT_LOCAL);
data_rtx_lsh = op_stat_reg ("RTX.Data RTX Rxd", OPC_STAT_INDEX_NONE,
    OPC_STAT_LOCAL);
//energy stats
energy_lsh = op_stat_reg ("Energy.Badge Energy (J)",
    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
sleep_time_lsh = op_stat_reg ("Energy.Sleep Time (secs)",
    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
tx_time_lsh = op_stat_reg ("Energy.Tx Time (secs)",
    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
rx_time_lsh = op_stat_reg ("Energy.Rx Time (secs)",
    OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
//global pkt stats
RTS_gsh = op_stat_reg ("Badge.RTS Sent", OPC_STAT_INDEX_NONE,
    OPC_STAT_GLOBAL);
CTS_gsh = op_stat_reg ("Badge.CTS Rxd", OPC_STAT_INDEX_NONE,
    OPC_STAT_GLOBAL);
DATA_gsh = op_stat_reg ("Badge.Data Sent", OPC_STAT_INDEX_NONE,
    OPC_STAT_GLOBAL);
```

```

ACK_gsh = op_stat_reg ("Badge.ACK Rxd", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
death_gsh = op_stat_reg ("Badge.Time Of Death", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
//global rtx stats
cts_rtx_gsh = op_stat_reg ("RTX.CTS RTX Sent", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
ack_rtx_gsh = op_stat_reg ("RTX.ACK RTX Sent", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
rts_rtx_gsh = op_stat_reg ("RTX.RTS RTX Rxd", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
data_rtx_gsh = op_stat_reg ("RTX.Data RTX Rxd", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
//global energy stats
energy_gsh = op_stat_reg ("Energy.Energy", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
sleep_time_gsh = op_stat_reg ("Energy.Sleep Time",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);
tx_time_gsh = op_stat_reg ("Energy.TX Time", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
rx_time_gsh = op_stat_reg ("Energy.RX Time", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);

//get ID
op_ima_obj_attr_get_int32(op_id_self(), "badge_ID", &my_ID);

//set range
op_ima_obj_attr_get_int32(op_id_self(), "range", &my_range);
switch(my_range) {
    case 3:
        my_threshold = RANGE_3M;
        break;
    case 4:
        my_threshold = RANGE_4M;
        break;
    case 5:
        my_threshold = RANGE_5M;
        TX_POWER = TX_POWER_5;
        break;
    case 6:
        my_threshold = RANGE_6M;
        TX_POWER = TX_POWER_6;
        break;
    default:
        my_threshold = RANGE_5M;
        break;
}

//schedule first wakeup at random time < 2 minutes
random_sleep();

//enable time logging
log_time = 1;

//set energy
energy = INITIAL_ENERGY;
rts_pkts_tx = ret_pkts_tx = data_pkts_tx = 0;

//initialise seq_no
RTS_seq = 0;
data_seq = 0;

```


D.1.2 INIT Exit Executives

```
intrpt_code = op_intrpt_code ();
```

D.1.3 SLEEP Enter Executives

```
//stay in sleep state until a wake_up interrupt is generated
```

```
//energy tracking
if(log_time) {
    enter_sleep = op_sim_time();
    if(rts_pkts_tx > 0) { //weve awoken and returned to sleep
        tx_time = (rts_pkts_tx * ACK_PKT_DELAY) + (ret_pkts_tx *
RET_PKT_DELAY) + \
                    (data_pkts_tx * DATA_PKT_DELAY);
        op_stat_write(tx_time_lsh, tx_time);
        op_stat_write(tx_time_gsh, tx_time);
        //rx time is remainder time out of sleep
        rx_time = enter_sleep - exit_sleep - tx_time;
        op_stat_write(rx_time_lsh, rx_time);
        op_stat_write(rx_time_gsh, rx_time);
        //new energy
        energy = energy - tx_time*TX_POWER - rx_time*RX_POWER;
        op_stat_write(energy_lsh, energy);
        op_stat_write(energy_gsh, energy);
        if(energy <= 0) {
            energy_evh = op_intrpt_schedule_self (op_sim_time
(), NO_ENERGY);
        }
    }
    //reset counters
    rts_pkts_tx = ret_pkts_tx = data_pkts_tx = 0;
}
```

D.1.4 SLEEP Exit Executives

```
intrpt_code = op_intrpt_code ();
//set CTS_count
CTS_count = 0;
```

D.1.5 RX_CTS Exit Executives

```
intrpt_code = op_intrpt_code ();
//interrupt received, determine what it is
CTS_code = check_CTS();
```

D.1.6 RX_ACK Enter Executives

```
//set timeout for ACK
timeout_evh = op_intrpt_schedule_self (op_sim_time () + TIMEOUT_TIME,
TIMEOUT); //time in seconds
```

D.1.7 RX_ACK Exit Executives

```
intrpt_code = op_intrpt_code ();

ACK_code = check_ACK();
```

D.1.8 Header Block

```
#include <math.h>
#include <oms_dist_support.h> //for distributions

//update interval
```

```

#define PKT_INTRVL      120 //seconds

//threshold range values
#define RANGE_3M -76.9848281491
#define RANGE_4M -79.4836028813
#define RANGE_5M -81.4218031415
#define RANGE_6M -83.0054280624

//CTS retries
#define MAX_CTS_RETRIES 3

//packet types
#define BADGE_RTS 1
#define BADGE_CTS 2
#define BADGE_ACK 3

/* Self Interrupt code values.*/
#define WAKE_UP      1
#define TIMEOUT      2
#define NO_ENERGY    3

//CTS intrpt types
#define CTS_VALID_CTS 0
#define CTS_RTS      1
#define CTS_OTHER_CTS 2
#define CTS_TIMEOUT  3
#define CTS_RETRANS  4

//ACK intrpt types
#define ACK_RECEIVED  0
#define ACK_TIMEOUT  1
#define ACK_RETRANS  2

/* Macro definitions for state transitions.      */
#define AWAKEN        (intrpt_code == WAKE_UP)
#define TIMEOUT_INT (intrpt_code == TIMEOUT)

#define VALID_CTS      (CTS_code == CTS_VALID_CTS)
#define RTS_RXD        (CTS_code == CTS_RTS)
#define CTS_RXD        (CTS_code == CTS_OTHER_CTS)
#define NO_CTS         (CTS_code == CTS_TIMEOUT &&
CTS_count < MAX_CTS_RETRIES)
#define NO_CTS_SLEEP   (CTS_code == CTS_TIMEOUT &&
CTS_count >= MAX_CTS_RETRIES)
#define WAIT_RETRANS   (CTS_code == CTS_RETRANS)

#define ACK_RXD        (ACK_code == ACK_RECEIVED)
#define NO_ACK_RXD     (ACK_code == ACK_TIMEOUT)
#define ACK_RTX        (ACK_code == ACK_RETRANS)

//timeout value (no ACK rxd)
#define TIMEOUT_TIME    0.05 //50ms

//time to wait if receive a RTS or CTS
#define ACK_PKT_DELAY   0.0010416833
#define DATA_PKT_DELAY 0.0141666833
#define RET_PKT_DELAY   0.0010416833
#define RTS_SLEEP       (2*ACK_PKT_DELAY + DATA_PKT_DELAY
+ 0.001) //cts+data+ack to go
#define CTS_SLEEP       (ACK_PKT_DELAY + DATA_PKT_DELAY +
0.001) //data+ack to go

```

```

//energy constants
#define INITIAL_ENERGY      2376 //Joules (220mAh CR2032)
#define SLEEP_POWER 0.078 //Watts
#define RX_POWER            0.0912
#define TX_POWER_5         0.0825
#define TX_POWER_6         0.0837

#define ZERO_ENERGY (op_intrpt_type() == OPC_INTRPT_SELF &&
intrpt_code == NO_ENERGY)

//sequencing
#define MAX_SEQ_NO          63
#define SEQ_INTRVL          1

//create my own TDA index
#define OPC_TDA_RA_PK_RETRANS (OPC_TDA_RA_MAX_INDEX + 1)

//function prototypes
static void gen_RTS(void);
static void clear_strm(void);
static int check_threshold(Packet* ptr);
static int checkErrorCTS(Packet* ptr, int id);
static int check_CTS(void);
static void wait_for_medium(void);
static void random_sleep(void);
static void proc_CTS_gen_data(void);
static int checkErrorACK(Packet* ptr, int id);
static int check_ACK(void);
static void process_ACK(void);

```

D.1.9 Function Block

```

//create an RTS packet
static void gen_RTS(void) {
    //just awoken so schedule next wake up
    wakeup_evh = op_intrpt_schedule_self (op_sim_time () +
PKT_INTRVL, WAKE_UP); //time in seconds
    //set awake stat
    op_stat_write(awake_lsh, 1.0);

    RTS_pkt = op_pk_create_fmt("badge_ACK");
    op_pk_nfd_set(RTS_pkt, "badge_id", my_ID);
    op_pk_nfd_set(RTS_pkt, "type", BADGE_RTS);
    op_pk_nfd_set(RTS_pkt, "seq_no", RTS_seq);

    RTS_seq+= SEQ_INTRVL;
    if(RTS_seq > MAX_SEQ_NO) {
        RTS_seq = RTS_seq - (MAX_SEQ_NO + 1);
    }

    /* Update the RTS generation statistics. */
    op_stat_write (RTSs_sent_lsh, 1.0);
    op_stat_write (RTS_gsh, 1.0);

    //copy pkt in case we have to RTX
    cp_ptr = op_pk_copy(RTS_pkt);

    //record sleep time, and update energy
    op_stat_write(sleep_time_lsh, op_sim_time() - enter_sleep);
    op_stat_write(sleep_time_gsh, op_sim_time() - enter_sleep);

```

```

        energy = energy - (SLEEP_POWER * (op_sim_time() -
enter_sleep));
        op_stat_write(energy_lsh, energy);
        op_stat_write(energy_gsh, energy);
        exit_sleep = op_sim_time();
        rts_pkts_tx++;
        //enable time logging
        log_time = 1;

        /* Send the packet via the stream to the lower layer. */
        if(energy > 0) {
            op_pk_send (RTS_pkt, 0);
        } else {
            energy_evh = op_intrpt_schedule_self (op_sim_time (),
NO_ENERGY);
        }

        //reset CTS timeout counter
        CTS_count = 0;
        //set timeout
        timeout_evh = op_intrpt_schedule_self (op_sim_time () +
TIMEOUT_TIME, TIMEOUT); //time in seconds

        //clear retrans_sent
        retrans_sent = OPC_FALSE;
        cp_pkt_sent = OPC_FALSE;
    }

static void clear_strm(void) {
    Packet* pkptr;
    //were asleep so any pkts need to be destroyed as our radio is
off in theory
    if(op_intrpt_type() == OPC_INTRPT_STRM) { //new pkt
        pkptr = op_pk_get (op_intrpt_strm ());
        op_pk_destroy(pkptr);
    }
    //disable time logging
    log_time = 0;
}

//check threshold of incoming packet to verify it should exist
static int check_threshold(Packet* ptr) {
    double thres_pwr;

    //Calc received power
    // dBm = 10log(mW)
    thres_pwr = 0.5 * op_td_get_dbl(ptr, OPC_TDA_RA_RCVD_POWER);
    thres_pwr = 1000 * thres_pwr; //mW
    thres_pwr = 10*log10(thres_pwr); //dBm
    op_stat_write (threshold_lsh, thres_pwr);

    //and check if over threshold
    if ( thres_pwr >= my_threshold) {
        return OPC_TRUE;
    } else {
        return OPC_FALSE;
    }
}

static int checkErrorCTS(Packet* ptr, int id) {
    if(op_td_get_int(ptr, OPC_TDA_RA_PK_RETRANS) ) { //error free

```

```

        //destroy backup packet
        if(!cp_pkt_sent) {
            op_stat_write(bob_gsh, 1.0);
            op_pk_destroy(cp_ptr);
        }
        return CTS_VALID_CTS;
    } else if(!retrans_sent) {
        //send a re-transmit request
        retrans_pkt = op_pk_create_fmt("badge_retrans");
        op_pk_nfd_set(retrans_pkt, "dest_id", 0);
        op_pk_nfd_set(retrans_pkt, "src_id", my_ID);
        op_pk_send(retrans_pkt, 0);
        op_stat_write(cts_rtx_lsh, 1.0);
        op_stat_write(cts_rtx_gsh, 1.0);
        ret_pkts_tx++;
        retrans_sent = OPC_TRUE;
        return CTS_RETRANS;
    } else {
        return CTS_TIMEOUT;
    }
}

//determine if intrpt is a valid CTS reply
static int check_CTS(void) {
    //intrpt here should either be new_pkt, or timeout
    char format[20];
    Packet* pkpPtr;
    int pkt_id;
    int type;
    int ret_code;

    if(op_intrpt_type() == OPC_INTRPT_STRM) { //new pkt
        /* Obtain the incoming packet. */
        pkpPtr = op_pk_get (op_intrpt_strm ());
        op_pk_format(pkpPtr, format);
        //check if over threshold
        if( check_threshold(pkpPtr) ) { //pkt exists
            //find packet type
            if(strcmp(format, "badge_retrans") == 0    &&
!cp_pkt_sent) { //retransmit packet received
                //check its for us
                op_pk_nfd_get(pkpPtr, "dest_id", &pkt_id);
                if(pkt_id == my_ID) {
                    //since we were expecting a CTS we must resend the
last RTS

                    op_pk_send(cp_ptr, 0);
                    cp_pkt_sent = OPC_TRUE;
                    rts_pkts_tx++;
                    //update stats
                    op_stat_write(rts_rtx_lsh, 1.0);
                    op_stat_write(rts_rtx_gsh, 1.0);
                }
                ret_code = -1;
            } else if(strcmp(format, "badge_ACK") == 0 ) {
//ACK fmt packet received
                //get id, check the pkt is for us
                op_pk_nfd_get(pkpPtr, "badge_id", &pkt_id);
                op_pk_nfd_get(pkpPtr, "type", &type);
                if( (pkt_id == my_ID) && (type == BADGE_CTS)
) { //its our CTS

                    //check for errors

```

```

        ret_code = checkErrorCTS(pkptr,
pkt_id);
        } else if (type == BADGE_CTS) { //another CTS
            ret_code = CTS_OTHER_CTS;
        } else if (type == BADGE_RTS) { //another RTS
            if(pkt_id == my_ID) { //hearing our own
tx
                ret_code = -1;
            } else {
                ret_code = CTS_RTS;
            }
        } else {
            ret_code = -1;
        }
    } else {
        ret_code = -1;
    }
} //check threshold
//destroy pkt
op_pk_destroy (pkptr);
return ret_code;
} else if (op_intrpt_type() == OPC_INTRPT_SELF && intrpt_code ==
TIMEOUT) { //timeout
    CTS_count++;
    //reset timeout
    timeout_evh = op_intrpt_schedule_self (op_sim_time () +
TIMEOUT_TIME, TIMEOUT); //time in seconds
    return CTS_TIMEOUT;
} else {
    return -1;
}
}

static void wait_for_medium(void) {
    //go to sleep since medium is busy
    random_wait = op_dist_uniform(0.01);
    if(RTS_RXD) {
        if (op_ev_valid(wakeup_evh) == OPC_TRUE) {
            op_ev_cancel(wakeup_evh);
        }
        wakeup_evh = op_intrpt_schedule_self (op_sim_time () +
random_wait + RTS_SLEEP, WAKE_UP); //time in seconds
    } else if (CTS_RXD) {
        if (op_ev_valid(wakeup_evh) == OPC_TRUE) {
            op_ev_cancel(wakeup_evh);
        }
        wakeup_evh = op_intrpt_schedule_self (op_sim_time () +
random_wait + CTS_SLEEP, WAKE_UP); //time in seconds
    }
    //no CTS coming, so fix seq counter
    RTS_seq -= SEQ_INTRVL;
    if(RTS_seq < 0) {
        RTS_seq = RTS_seq + (MAX_SEQ_NO+1);
    }
    //adjust RTS gen stat
    op_stat_write (RTSs_sent_lsh, -1.0);
    op_stat_write (RTS_gsh, -1.0);
}

static void random_sleep(void) {

```

```

    //giving up on CTS so fix seq counter
    RTS_seq-= SEQ_INTRVL;
    if(RTS_seq < 0) {
        RTS_seq = RTS_seq + (MAX_SEQ_NO+1);
    }
    //cancel last wakeup and reschedule
    if (op_ev_valid(wakeup_evh) == OPC_TRUE) {
        op_ev_cancel(wakeup_evh);
    }
    wake_time = op_dist_uniform(PKT_INTRVL);
    wakeup_evh = op_intrpt_schedule_self (op_sim_time () +
wake_time, WAKE_UP); //time in seconds
}

static void proc_CTS_gen_data(void) {
    //log CTS
    op_stat_write(CTS_lsh, 1.0);
    op_stat_write(CTS_gsh, 1.0);
    //cancel timeout
    if (op_ev_valid(timeout_evh) == OPC_TRUE) {
        op_ev_cancel(timeout_evh);
    }
    //generate a data packet and await ACK
    data_pkt = op_pk_create_fmt("badge_data");
    op_pk_nfd_set(data_pkt, "badge_id", my_ID);
    op_pk_nfd_set(data_pkt, "seq_no", data_seq);
    data_seq+= SEQ_INTRVL;
    if(data_seq > MAX_SEQ_NO) {
        data_seq = data_seq - (MAX_SEQ_NO+1);
    }

    /* Update the packet generation statistics.*
    op_stat_write (pkts_sent_lsh, 1.0);
    op_stat_write(DATA_gsh, 1.0);

    //copy pkt in case rtx
    cp_ptr = op_pk_copy(data_pkt);

    /* Send the packet via the stream to the lower layer.*
    op_pk_send (data_pkt, 0);

    //energy accounting
    data_pkts_tx++;

    //reset RTX
    retrans_sent = OPC_FALSE;
    cp_pkt_sent = OPC_FALSE;
}

static int checkErrorACK(Packet* ptr, int id) {
    if(op_td_get_int(ptr, OPC_TDA_RA_PK_RETRANS)) { //error free
        //destroy backup packet
        if(!cp_pkt_sent) {
            op_pk_destroy(cp_ptr);
        }
        return ACK_RECEIVED;
    } else if(!retrans_sent) {
        //send a re-transmit request
        retrans_pkt = op_pk_create_fmt("badge_retrans");
        op_pk_nfd_set(retrans_pkt,"dest_id", 0);
        op_pk_nfd_set(retrans_pkt,"src_id", my_ID);
    }
}

```

```

        op_pk_send(retrans_pkt, 0);
        op_stat_write(ack_rtx_lsh, 1.0);
        op_stat_write(ack_rtx_gsh, 1.0);
        ret_pkts_tx++;
        retrans_sent = OPC_TRUE;
        return ACK_RETRANS;
    } else {
        return ACK_TIMEOUT;
    }
}

//determine if intrpt is a valid ACK reply
static int check_ACK(void) {
    char format[20];
    Packet* pkptr;
    int pkt_id;
    int type;
    int ret_code;

    if(op_intrpt_type() == OPC_INTRPT_STRM) { //new pkt
        /* Obtain the incoming packet. */
        pkptr = op_pk_get (op_intrpt_strm ());
        op_pk_format(pkptr, format);
        //check if over threshold
        if( check_threshold(pkptr) ) { //pkt exists
            //find packet type
            if(strcmp(format, "badge_retrans") == 0 &&
!cp_pkt_sent) { //retransmit packet received
                //check its for us
                op_pk_nfd_get(pkptr, "dest_id", &pkt_id);
                if(pkt_id == my_ID) {
                    //since we were expecting a ACK we must resend the
last data
                        op_pk_send(cp_ptr, 0);
                        cp_pkt_sent = OPC_TRUE;
                        data_pkts_tx++;
                        op_stat_write(data_rtx_lsh, 1.0);
                        op_stat_write(data_rtx_gsh, 1.0);
                }
                ret_code = -1;
            } else if(strcmp(format, "badge_ACK") == 0 ) {
//ACK fmt packet received
                //get id, check the pkt is for us
                op_pk_nfd_get(pkptr, "badge_id", &pkt_id);
                op_pk_nfd_get(pkptr, "type", &type);
                if( (pkt_id == my_ID) && (type == BADGE_ACK)
) { //its our ACK
                    //check for errors
                    ret_code = checkErrorACK(pkptr,
pkt_id);
                } else {
                    ret_code = -1;
                }
            } else {
                ret_code = -1;
            }
        } //check threshold
        //destroy pkt
        op_pk_destroy (pkptr);
        return ret_code;
    }
}

```



```

        } else if(op_intrpt_type() == OPC_INTRPT_SELF && intrpt_code ==
TIMEOUT) { //timeout
            return ACK_TIMEOUT;
        } else {
            return -1;
        }
    }

static void process_ACK(void) {
    //log ACK
    op_stat_write(ACK_lsh, 1.0);
    op_stat_write(ACK_gsh, 1.0);
    //cancel timeout
    if (op_ev_valid(timeout_evh) == OPC_TRUE) {
        op_ev_cancel(timeout_evh);
    }
}

static void logDeath(void) {
    //node dead, log time of death
    op_stat_write(death_gsh, op_sim_time());
}

```

D.2 Base Station

For the base station there is code in the Enter Executives of the INIT state and the Exit Executives of the FORWARD state. There is also code in the Header and Function blocks as well as the Temporary Variables.

D.2.1 INIT Enter Executives

```

/* Initilaize the statistic handles to keep      */
/* track of traffic sinked by this process.      */
threshold_lsh = op_stat_reg ("Rx Power (dBm)", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
data_lsh = op_stat_reg ("Data Pkts Rxd", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
RTS_lsh = op_stat_reg ("RTS Pkts Rxd", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
timeout_lsh = op_stat_reg ("Data Timeout", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
delay_gsh = op_stat_reg ("End-to-End Delay (seconds)",
OPC_STAT_INDEX_NONE, OPC_STAT_GLOBAL);

//set range
op_ima_obj_attr_get_int32(op_id_self(), "range", &my_range);
switch(my_range) {
    case 3:
        my_threshold = RANGE_3M;
        break;
    case 4:
        my_threshold = RANGE_4M;
        break;
    case 5:
        my_threshold = RANGE_5M;
        break;
    case 6:
        my_threshold = RANGE_6M;
        break;
}

```

```

        default:
            my_threshold = RANGE_5M;
            break;
    }

    //not waiting for data so free to accept RTS
    free = OPC_TRUE;
    //no copied pkt to destroy
    destroy = OPC_FALSE;
    //no cancel rxd yet
    cancel = OPC_FALSE;

```

D.2.2 FORWARD Exit Executives

```

char  format[20];
int    ret_id;
Packet* retrans_pkt;

intrpt_code = op_intrpt_code();

if(op_intrpt_type() == OPC_INTRPT_STRM) {
    /* Obtain the incoming packet.      */
    pkptr = op_pk_get (op_intrpt_strm ());
    op_pk_format(pkptr, format);

    //check stream
    if(op_intrpt_strm() == MOBILE) {
        //check if over threshold
        if ( check_threshold(pkptr) ) { //pkt exists
            if(strcmp(format, "badge_retrans") == 0) {
                //check id
                op_pk_nfd_get(pkptr, "src_id", &ret_id);
                if(pkt_id == ret_id && pkt_id != 0 &&
!cancel) {
                    //retrans request, resend last packet
                    op_pk_send(cp_ptr, MOBILE);
                    retrans_sent = OPC_FALSE;
                    destroy = OPC_FALSE;
                }
                op_pk_destroy(pkptr);
            } else if( checkError(pkptr) ) {
                //error free, continue
                if(strcmp(format, "badge_ACK") == 0) {
                    op_pk_nfd_get(pkptr, "type", &type);

                    if(type == BADGE_RTS && free ==
OPC_TRUE) {
                        cancel = OPC_FALSE;
                        //destroy copied pkt
                        if(destroy) {
                            op_pk_destroy(cp_ptr);
                        }
                        //save id to SV so can send thru on timeout
                        op_pk_nfd_get(pkptr, "badge_id",
&pkt_id);

                        //forward to server
                        op_pk_send(pkptr, WIRED);
                        retrans_sent = OPC_FALSE;
                        //log reception
                        op_stat_write(RTS_lsh, 1.0);

```

```

                                op_stat_write(delay_gsh,
op_sim_time() - op_pk_creation_time_get(pkptr));
                                //now wait for data or timeout
                                free = OPC_FALSE;
                                data_sent = OPC_FALSE;
                                //set timeout
                                timeout_evh =
op_intrpt_schedule_self(op_sim_time() + DATA_TIMEOUT, TIMEOUT_INT);
                                } else {
                                    op_pk_destroy(pkptr);
                                }
                                } else if(strcmp(format, "badge_data") == 0)
{
                                //destroy copied pkt
                                if(destroy) {
                                    op_pk_destroy(cp_ptr);
                                }
                                //data packet received, forward
                                op_pk_send(pkptr, WIRED);
                                retrans_sent = OPC_FALSE;
                                //log reception
                                op_stat_write(data_lsh, 1.0);
                                op_stat_write(delay_gsh, op_sim_time()
- op_pk_creation_time_get(pkptr));
                                data_sent = OPC_TRUE;
                                //cancel timeout
                                if(op_ev_valid(timeout_evh)) {
                                    op_ev_cancel(timeout_evh);
                                }
                                }
                                }else { //has errors, send a RTX
                                op_stat_write(base_gsh, 1.0);
                                if(!retrans_sent) {
                                    //send a re-transmit request
                                    op_pk_nfd_get(pkptr, "badge_id",
&ret_id);
                                    retrans_pkt =
op_pk_create_fmt("badge_retrans");
                                    op_pk_nfd_set(retrans_pkt, "dest_id",
ret_id);
                                    op_pk_nfd_set(retrans_pkt, "src_id", 0);
                                    op_pk_send(retrans_pkt, MOBILE);
                                    // op_stat_write(ack_rtx_lsh, 1.0);
                                    // op_stat_write(ack_rtx_gsh, my_ID);
                                    retrans_sent = OPC_TRUE;
                                }
                                }
                                } else { //under threshold
                                /* Destroy the received packet.      */
                                op_pk_destroy (pkptr);
                                }
                                } else if(op_intrpt_strm() == WIRED) {
                                if(strcmp(format, "badge_ACK") == 0) {
                                    op_pk_nfd_get(pkptr, "type", &type);
                                    op_pk_nfd_get(pkptr, "badge_id", &pkt_id);
                                    cp_ptr = op_pk_copy(pkptr);
                                    destroy = OPC_TRUE; //copied pkt must be destroyed
if not sent
                                    if(type == BADGE_CTS) {
                                        op_pk_send(pkptr, MOBILE);
                                    } else if(type == BADGE_ACK) {

```

```

        op_pk_send(pkptr, MOBILE);
        //transfer finished so free again
        free = OPC_TRUE;
    } else if(type == BADGE_CANCEL) {
        //another base already handling this so just destroy
        op_pk_destroy(pkptr);
        //we must keep silent until badge and other base
        //have finished sending data
        if(data_sent) {
            //we can accept next once weve heard the data
            free = OPC_TRUE;
        }
        cancel = OPC_TRUE;
    }
}

} //end intrpt_strm
} else if(op_intrpt_type() == OPC_INTRPT_SELF && intrpt_code ==
TIMEOUT_INT) {
    //data timeout, set base free again
    free = OPC_TRUE;
    //log timeout
    op_stat_write(timeout_lsh, 1.0);
    //send cancel to server to decrement sequence number
    newpkt = op_pk_create_fmt("badge_ACK");
    op_pk_nfd_set(newpkt, "badge_id", pkt_id);
    op_pk_nfd_set(newpkt, "type", BADGE_CANCEL);
    op_pk_send(newpkt, WIRED);
}

```

D.2.3 Header Block

```

#include <math.h>

//threshold range values
#define RANGE_3M -76.9848281491
#define RANGE_4M -79.4836028813
#define RANGE_5M -81.4218031415
#define RANGE_6M -83.0054280624

//packet types
#define BADGE_RTS      1
#define BADGE_CTS      2
#define BADGE_ACK      3
#define BADGE_CANCEL   0

//stream definitions
#define MOBILE         0
#define WIRED          1

//intrpt
#define DATA_TIMEOUT   0.5
#define TIMEOUT_INT     1

//create my own TDA index
#define OPC_TDA_RA_PK_RETRANS (OPC_TDA_RA_MAX_INDEX + 1)

//function prototypes
static int check_threshold(Packet* ptr);
static int checkError(Packet* ptr);

```

D.2.4 Function Block

```
static int check_threshold(Packet* ptr) {
    double thres_pwr;

    //Calc received power
    // dBm = 10log(mW)
    thres_pwr = 0.5 * op_td_get_dbl(ptr, OPC_TDA_RA_RCVD_POWER);
    thres_pwr = 1000 * thres_pwr; //mW
    thres_pwr = 10*log10(thres_pwr); //dBm
    op_stat_write (threshold_lsh, thres_pwr);

    //and check if over threshold
    if ( thres_pwr >= my_threshold) {
        return OPC_TRUE;
    } else {
        return OPC_FALSE;
    }
}

static int checkError(Packet* ptr) {
    if(op_td_get_int(ptr, OPC_TDA_RA_PK_RETRANS)) { //error free
        return OPC_TRUE;
    } else {
        return OPC_FALSE;
    }
}
```

D.2.5 Temporary Variables

```
Packet*      pkptr;
Packet*      newpkt;
int          type;
```

D.3 CSMA

The CSMA process is in both the SmartBadge and base station node models as the “tx_proc” block and it is shown in Figure D.1. There is code written for the Enter Executives of all three states, as well as the Header Block.

D.3.1 idle Enter Executives

```
wait_lsh = op_stat_reg("Waiting", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
wait_gsh = op_stat_reg("Waiting", OPC_STAT_INDEX_NONE,
OPC_STAT_GLOBAL);
```

D.3.2 wt_free Enter Executives

```
op_stat_write(wait_lsh, 1.0);
op_stat_write(wait_gsh, 1.0);
```

D.3.3 tx_pkt Enter Executives

```
/* Outgoing packet */
Packet *out_pkt;
/* A packet has arrived for transmission. Acquire */
/* the packet from the input stream and send the packet out.*/
out_pkt = op_pk_get (IN_STRM);
op_pk_send (out_pkt, OUT_STRM);
```

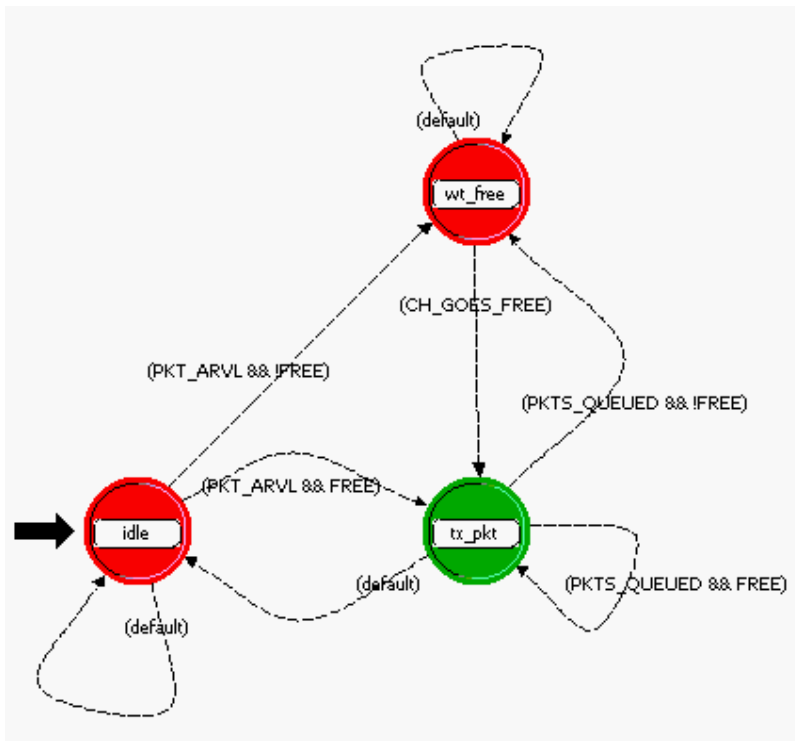


Figure D.1 CSMA Process Model

D.3.4 Header Block

```

/* Input stream from generator module */
#define IN_STRM 0
/* Output stream to bus transmitter module */
#define OUT_STRM 0
/* Conditional macros */
#define PKT_ARVL (op_intrpt_type() == OPC_INTRPT_STRM)

/* input statistic indices */
#define CH_BUSY_STAT 0
/* Conditional macros */
#define FREE (op_stat_local_read (CH_BUSY_STAT) == 0.0)
#define PKTS_QUEUED (!op_strm_empty (IN_STRM))
#define CH_GOES_FREE (op_intrpt_type () == OPC_INTRPT_STAT)

```

D.4 Server

At the server code has been written for the Enter Executives of the INIT state and the Exit Executives of the REPLY state. There is also code in the Header Block and the Temporary Variables.

D.4.1 INIT Enter Executives

```

int i;
/* Initilaize the statistic handles to keep */
/* track of traffic sinked by this process. */
packet_lsh = op_stat_reg ("Data Pkts Rxd", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);
RTS_lsh = op_stat_reg ("RTS Pkts Rxd", OPC_STAT_INDEX_NONE,
OPC_STAT_LOCAL);

```

```

RTS_cancel_lsh = op_stat_reg ("RTS Cancels Sent",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);
data_cancel_lsh = op_stat_reg ("Data Cancels Sent",
OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL);

```

```

//initialise seq array with -1
for(i=0;i<256;i++) {
    seq[i][SEQ_RTS] = -1;
    seq[i][SEQ_DATA] = -1;
}

```

D.4.2 REPLY Exit Executives

```

char format[20];

```

```

if(op_intrpt_type() == OPC_INTRPT_STRM) {
    /* Obtain the incoming packet. */
    pkptr = op_pk_get(op_intrpt_strm());
    op_pk_format(pkptr, format);

    if(strcmp(format, "badge_ACK") == 0) {
        //extract type and ID
        op_pk_nfd_get(pkptr, "type", &type);
        op_pk_nfd_get(pkptr, "badge_id", &pkt_id);
        op_pk_nfd_get(pkptr, "seq_no", &seq_no);
        if(type == BADGE_RTS) {
            //check we are the first to receive this pkt
            if( (seq[pkt_id][SEQ_RTS] < seq_no) || (seq_no == 0 &&
seq[pkt_id][SEQ_RTS] == MAX_SEQ_NO) ) {
                //new pkt so process
                op_stat_write(RTS_lsh, 1.0); //pkt_id);
                //update seq_no rxd
                seq[pkt_id][SEQ_RTS] = seq_no;
                //send CTS reply
                newpkt = op_pk_create_fmt("badge_ACK");
                op_pk_nfd_set(newpkt, "badge_id", pkt_id);
                op_pk_nfd_set(newpkt, "type", BADGE_CTS);
                op_pk_send(newpkt, op_intrpt_strm()); //output
on same stream
            } else {
                //send cancel pkt to base station
                newpkt = op_pk_create_fmt("badge_ACK");
                op_pk_nfd_set(newpkt, "badge_id", 0);
                op_pk_nfd_set(newpkt, "type", BADGE_CANCEL);
                op_pk_send(newpkt, op_intrpt_strm()); //output
on same stream

                op_stat_write(RTS_cancel_lsh, 1.0); //pkt_id);
            }
            op_pk_destroy(pkptr);
        } else if (type == BADGE_CANCEL) {
            //base has timed out on data, adjust seq number
            seq[pkt_id][SEQ_RTS] -= SEQ_INTRVL;
            if(seq[pkt_id][SEQ_RTS] < 0) {
                seq[pkt_id][SEQ_RTS] += (MAX_SEQ_NO+1);
            }
        } //end RTS
    } else if(strcmp(format, "badge_data") == 0) {
        //data packet received
        //read id, then destroy pkt
        op_pk_nfd_get(pkptr, "badge_id", &pkt_id);
        op_pk_nfd_get(pkptr, "seq_no", &seq_no);

```

```

        op_pk_destroy (pkptr);
        //check we are the first to receive this pkt
        if( (seq[pkt_id][SEQ_DATA] < seq_no) || (seq_no == 0 &&
seq[pkt_id][SEQ_DATA] == MAX_SEQ_NO) ) {
            //new pkt so process
            op_stat_write(packet_lsh, 1.0);//pkt_id); //store
data ie log pkts rxd
            //update seq_no rxd
            seq[pkt_id][SEQ_DATA] = seq_no;
            //send ACK to id
            newpkt = op_pk_create_fmt("badge_ACK");
            op_pk_nfd_set(newpkt, "badge_id", pkt_id);
            op_pk_nfd_set(newpkt, "type", BADGE_ACK);
            op_pk_send(newpkt,op_intrpt_strm()); //output on
same stream
        } else {
            //send cancel pkt to base station
            newpkt = op_pk_create_fmt("badge_ACK");
            op_pk_nfd_set(newpkt, "badge_id", 0);
            op_pk_nfd_set(newpkt, "type", BADGE_CANCEL);
            op_pk_send(newpkt,op_intrpt_strm()); //output on
same stream
            op_stat_write(data_cancel_lsh, 1.0);//100*pkt_id);
        }
    } else {
        /* Destroy the received packet.      */
        op_pk_destroy (pkptr);
    }
}

```

D.4.3 Header Block

```

#include <math.h>

//threshold range values
#define RANGE_3M -76.9848281491
#define RANGE_4M -79.4836028813
#define RANGE_5M -81.4218031415
#define RANGE_6M -83.0054280624

//packet types
#define BADGE_RTS          1
#define BADGE_CTS          2
#define BADGE_ACK          3
#define BADGE_CANCEL      0

//sequence definitions
#define MAX_SEQ_NO         63
#define SEQ_RTS            0
#define SEQ_DATA           1
#define SEQ_INTRVL         1

```

D.4.4 Temporary Variables

```

Packet*      pkptr;
Packet*      newpkt;
int          pkt_id;
int          type;

```


D.5 Radio Pipeline

The final piece of code is the change we made to the “ecc” stage of the radio pipeline.

This stage decides whether to accept a packet depending on how many errors it has.

```
/* Modified error correction model for radio link */
/* Transceiver Pipeline. This model uses the receiver */
/* channel state information to update the signal lock */
/* status of the channel. It relies on the rxgroup stage */
/* model for the creation and initialization of the channel */
/* state information. */

#include "opnet.h"
#include "dra.h"

#ifdef __cplusplus
extern "C"
#endif

//create my own TDA index
#define OPC_TDA_RA_PK_RETRANS (OPC_TDA_RA_MAX_INDEX + 1)

void
badge_ecc_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
{
    int                num_errs, accept;
    OpT_Packet_Size    pklen;
    double             ecc_thresh;
    DraT_Rxch_State_Info* rxch_state_ptr;

    /** Determine acceptability of given packet at receiver. **/
    FIN_MT (badge_ecc (pkptr));

    /* Do not accept packets that were received */
    /* when the node was disabled. */
    if (op_td_is_set (pkptr, OPC_TDA_RA_ND_FAIL))
        accept = OPC_FALSE;
    else
    {
        /* Obtain the error correction threshold of the receiver. */
        ecc_thresh = op_td_get_dbl (pkptr, OPC_TDA_RA_ECC_THRESH);

        /* Obtain length of packet. */
        pklen = op_pk_total_size_get (pkptr);

        /* Obtain number of errors in packet. */
        num_errs = op_td_get_int (pkptr, OPC_TDA_RA_NUM_ERRORS);

        /* Test if bit errors exceed threshold. */
        if (pklen == 0)
            accept = OPC_TRUE;
        else
            accept = (((double) num_errs) / pklen) <=
ecc_thresh) ? OPC_TRUE : OPC_FALSE;
    }

    //always set flag to accept pkt
    op_td_set_int (pkptr, OPC_TDA_RA_PK_ACCEPT, OPC_TRUE);
    //if errors exist set my own TDA value (TDA_RA_PK_RETRANS)
    //so that receiver can ask for a re-transmit (once)
    op_td_set_int(pkptr, OPC_TDA_RA_PK_RETRANS, accept);
}
```

```

        /* In either case the receiver channel is no longer locked. */
        rxch_state_ptr = (DraT_Rxch_State_Info *) op_ima_obj_state_get
(op_td_get_int (pkptr, OPC_TDA_RA_RX_CH_OBJID));
        rxch_state_ptr->signal_lock = OPC_FALSE;

FOUT
}

```

Appendix E Base Station Coverage

Area Derivations

Below are the equations used to calculate the coverage areas for the scenarios used in the OPNET simulations.

E.1 Scenario 1

This is the 10x10 metre room with a single base station (Figure E.).

The total room area is $10 \times 10 = 100 \text{ m}^2$

The base station has a range of 5m so it covers $\pi r^2 = \pi(5)^2 = 78.53982 \text{ m}^2$

The coverage is therefore $78.53981 / 100 = 78.54 \%$

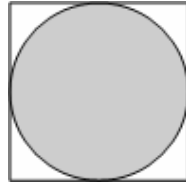


Figure E.1 Coverage area for scenario 1 (shaded)

E.2 Scenarios 2 and 5

Scenarios 2 and 5 are the 20 x 20 metre room with 4 base stations and range of 5m (Figure E.2). This is just four of the Scenario 1 room merged into a larger room and so has the same coverage percentage, i.e. 78.54 %.

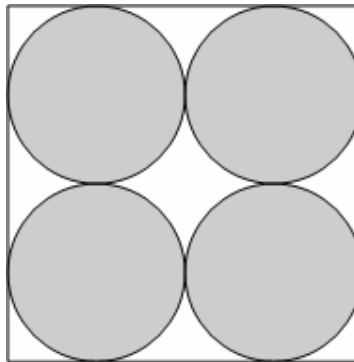


Figure E.2 Coverage area for scenarios 2 and 5 (shaded)

E.3 Scenarios 3 and 6

A 20m x 20m room with 4 base stations but a 6m range (Figure E.3).

The total area is $20 \times 20 = 400 \text{ m}^2$.

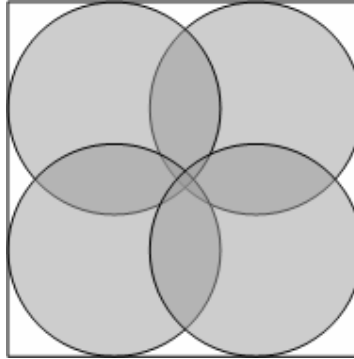


Figure E.3 Coverage area for scenarios 3 and 6 (shaded)

Each uncovered corner can be calculated by taking the space left from a 12m x 12m room with a 6m circle in the centre (shaded in Figure E.4). This area must be $\frac{1}{4}$ of the size of the square (12^2) less the size of the circle ($\pi 6^2$) i.e. $\frac{1}{4} (12^2 - \pi 6^2) = 7.725666118 \text{ m}^2$.

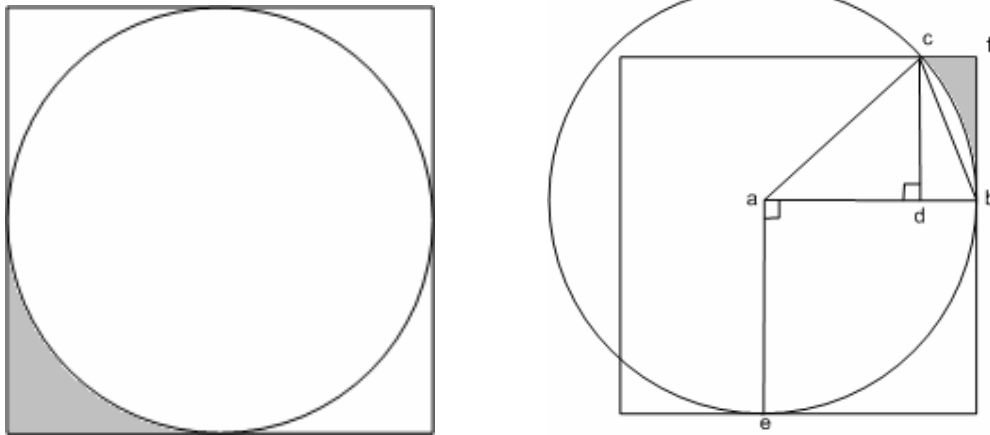


Figure E.4 Drawings for the uncovered corner and half uncovered edge

The uncovered edges are more difficult. Suppose we have a 10m x 10m square with a 6m radius circle (i.e. $\frac{1}{4}$ of the entire room, see the second square in Figure E.4). The shaded area is half of the uncovered edge region of scenarios 3 and 6. Lines ab, ac and ae are radii so are 6m long.

Line cd is 4m long since the square is 10m tall and the distance from the bottom of the square to ab is the same as the length ae (6m).

This gives line ad (by Pythagoras' theorem) a length of $\sqrt{6^2 - 4^2} = 4.472135955 \text{ m}$.

It follows that line bd (and so line cf) is $6 - 4.472135955 = 1.572864045 \text{ m}$.

We can also calculate angle cad , $\text{cad} = \sin^{-1}(\text{cd}/\text{ac}) = \sin^{-1}(4/6) = 0.7297276562 \text{ rad}$.

Now the area formed by the segment cut from the circle by line bc is $r^2[\theta - \sin(\theta)]/2 = 6^2[0.7297276562 - 4/6]/2 = 1.135097811 \text{ m}^2$.

Finally the area of the triangle bcf is $\frac{1}{2}(\text{bf} \times \text{cf}) = \frac{1}{2}(4 \times 1.572864045) = 3.05572809 \text{ m}^2$, and so the shaded area is $3.05572809 - 1.135097811 = 1.920630279 \text{ m}^2$.

The uncovered edge region is twice this i.e. 3.841260558 m^2 .

This gives the coverage area as the total area less 4 corners less 4 edges i.e. $400 - 4 \times 7.725666118 - 4 \times 3.841260558 = 353.7322933 \text{ m}^2$, which as a percentage of the total is 88.43 %.

E.4 Scenarios 4 and 7

This is again the 20m x 20m room, but with 5 base stations all with a 5m range (Figure E.5).

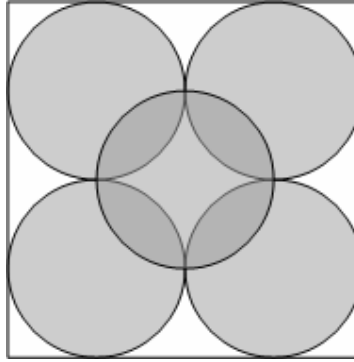


Figure E.5 Coverage area for scenarios 4 and 7 (shaded)

The four outer base stations cover the same area as in scenarios 2 and 6 i.e. $4 \times 78.53981634 = 314.1592654 \text{ m}^2$. The central base station adds to this the equivalent of four corners (if you divide the room into 10m x 10m squares) which must have an area equal to the uncovered part of scenario 1 or $100 - 78.53981634 = 21.46018366 \text{ m}^2$.

This gives a coverage area of $(314.1592654 + 21.46018366) / 400 = 83.90 \%$.

References

- [1] Poynting Software, "SuperNEC," 2.55 ed: <http://www.supernec.com/>.
- [2] Charmed Technology: <http://www.charmed.com/products/charmbadge.html>.
- [3] Bluetooth vs. Wi-Fi - Bluetooth website:
https://www.bluetooth.org/admin/bluetooth2/faq/view_record.php?id=49.
- [4] RFID Journal FAQs: <http://www.rfidjournal.com/faq>.
- [5] R. Borovoy, M. McDonald, F. Martin, and M. Resnick, "Things that blink: Computationally augmented name tags," *IBM Systems Journal*, vol. 35, pp. 488-495, 1996.
- [6] R. Borovoy, F. Martin, S. Vemuri, M. Resnick, B. Silverman, and C. Hancock, "Meme tags and community mirrors: moving from conferences to collaboration," in *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. Seattle, WA, 1998, pp. 159-168.
- [7] nTAG Interactive: http://www.ntag.com/downloads/nTAG_leaflet.pdf.
- [8] MIT Media Lab: <http://www.media.mit.edu/>.
- [9] nTAG Applications: <http://www.ntag.com/products/>.
- [10] Lancaster University Computing Department: <http://www.comp.lancs.ac.uk/>.
- [11] N. Villar, A. Schmidt, G. Kortuem, and H.-W. Gellersen, "Interacting with Proactive Community Displays," *Computers and Graphics*, vol. 27, 2003.
- [12] RF Technologies: <http://www.pinpointco.com/>.
- [13] Access Inc.: <http://www.axcessinc.com/>.
- [14] Varitronics Inc.: http://www.varitronics.com/var_ir.htm.
- [15] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," presented at INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2002.
- [16] IEEE 802.11 The Working Group Setting the Standards for Wireless LANs:
<http://grouper.ieee.org/groups/802/11/>.
- [17] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st international conference on*

Embedded networked sensor systems. Los Angeles, California, USA: ACM Press, 2003, pp. 171-180.

[18] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks," in *Proceedings of Algorithmic Aspects of Wireless Sensor Networks: First International Workshop, ALGOSENSORS 2004*, LNCS 3121 ed. Turku, Finland, 2004, pp. 18-31.

[19] A. El-Hoiydi, "Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks," in *Proceedings of the Seventh International Symposium on Computers and Communications, 2002 (ISCC 2002)*. Taormina, Italy, July 2002, pp. 685-692.

[20] H. Pham and S. Jha, "Addressing mobility in wireless sensor media access protocol," presented at Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004, Melbourne, Australia, December 2004.

[21] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," presented at 18th International Parallel and Distributed Processing Symposium, 2004 (IPDPS'04), Santa Fe, New Mexico, USA, April 2004.

[22] B. H. Liu, N. Bulusu, H. Pham, and S. Jha, "CSMAC: A novel DS-CDMA based MAC protocol for wireless sensor networks," presented at IEEE Global Telecommunications Conference Workshops, 2004 (GlobeCom Workshops 2004), Dallas, Texas, USA, November 2004.

[23] P. Lin, C. Qiao, and X. Wang, "Medium access control with a dynamic duty cycle for sensor networks," presented at IEEE Wireless Communications and Networking Conference, 2004 (WCNC 2004), Atlanta, Georgia, USA, March 2004.

[24] T. Zheng, S. Radhakrishnan, and V. Sarangan, "PMAC: An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," presented at 19th IEEE International Parallel and Distributed Processing Symposium, 2005 (IPDPS'05), Denver, Colorado, USA, April 2005.

[25] ECMA International, "Standard ECMA-340: Near Field Communication - Interface and Protocol (NFCIP-1)," <http://www.ecma-international.org/publications/standards/Ecma-340.htm>.

[26] W. H. M. Woo, "Wireless Local Area Network Using Near-Field RF Coupling," in *Department of Electrical and Computer Engineering*: University of Queensland, October 2001.

- [27] P. Neaves and J. Bedford-Roberts, "Dynamic Connection of Wearable Computers to Companion Devices Using Near-Field Radio," in *Proceedings of The Second International Symposium on Wearable Computers*, 1998, pp. 156-157.
- [28] R. A. Sainati, "CAD of microstrip antennas for wireless applications." Boston: Artech House, 1996, pp. 10.
- [29] The Mathworks, "MATLAB," R13 ed: <http://www.mathworks.com/>.
- [30] Double Quad "Bow Tie": <http://manuka.orcon.net.nz/wifibowt.gif>.
- [31] P. H. Young, "Electronic communication techniques," 5th ed. Upper Saddle River, N.J.: Pearson/Prentice Hall, 2004, pp. 191-194.
- [32] Chipcon, *CC1010: Single Chip Very Low Power RF Transceiver with 8051-Compatible Microcontroller*, 1.2 ed:
http://www.chipcon.com/files/CC1010_Data_Sheet_1_2.pdf, 2003.
- [33] TOIM4232 IR Endec Datasheet:
<http://www.vishay.com/product?docid=82546&query=toim4232>.
- [34] TDFU4100 IR Transceiver Datasheet: <http://www.vishay.com/ir-transceivers/list/product-82514/>.
- [35] NS-2 Network Simulator (ns-2.28): <http://www.isi.edu/nsnam/ns/>, 3 Feb 2005.
- [36] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, pp. 46-55, 1999.
- [37] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," in *Proceedings of the conference on Communications architectures, protocols and applications*. London, United Kingdom, 1994, pp. 234-244.
- [38] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," presented at IEEE Singapore International Conference on Networks (SICON'97), Singapore, April 1997.
- [39] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *ACM Mobile Networks and Applications Journal*, vol. 1, pp. 183-197, 1996.
- [40] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," presented at Second IEEE Workshop on Mobile Computing Systems and Applications, 1999 (WMCSA '99), New Orleans, Louisiana, USA, February 1999.
- [41] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile computing, Kluwer international series in engineering and*

computer science SECS 353, T. Imielinski and H. F. Korth, Eds. Boston: Kluwer Academic Publishers, 1996, pp. 153-181.

[42] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," presented at Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '97), Kobe, Japan, April 1997.

[43] M. S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," *ACM Wireless Networks Journal*, vol. 1, pp. 61-81, 1995.

[44] R. Dube, C. D. Rais, K.-Y. Wang, and S. K. Tripathi, "Signal stability-based adaptive routing (SSA) for ad hoc mobile networks," *IEEE Personal Communications*, vol. 4, pp. 36-45, 1997.

[45] OPNET Modeler: <http://www.opnet.com>.

[46] Ultralife Batteries: <http://www.ultralifebatteries.com>.

[47] U10004 Thin Cell Datasheet:

http://www.ultralifebatteries.com/documents/techsheets/UBI-5155_U10004.pdf.

[48] Chipcon, *CC2431: System-on-Chip for 2.4GHz ZigBee with Location Engine*: [http://www.chipcon.com/files/CC2431_Data_Sheet_rev1p00%20\(2\).pdf](http://www.chipcon.com/files/CC2431_Data_Sheet_rev1p00%20(2).pdf), 2005.

[49] 16x2 LCD Datasheet: [http://www.varitronix.com/catalog/lcm_spec/MDL\(S\)-16265XLV.pdf](http://www.varitronix.com/catalog/lcm_spec/MDL(S)-16265XLV.pdf).

[50] MM74HC164 Datasheet:

<http://www.fairchildsemi.com/ds/MM/MM74HC164.pdf>.

[51] Piezoelectric Buzzer PKLCS1212E4001-R1:

<http://search.murata.co.jp/Ceramy/owa/CATALOG.showcatalog?sHinnmTmp=PKLCS1212E4001-R1&sLang=2&sNhn=PKLCS1212E4001-R1&sHnTyp=OLD>.

[52] Red/Green LEDs LSGT670 Datasheet:

<http://www.farnell.com/datasheets/1884.pdf>.

[53] Pushbutton DTSM-32S-B Datasheet:

<http://www.farnell.com/datasheets/38311.pdf>.